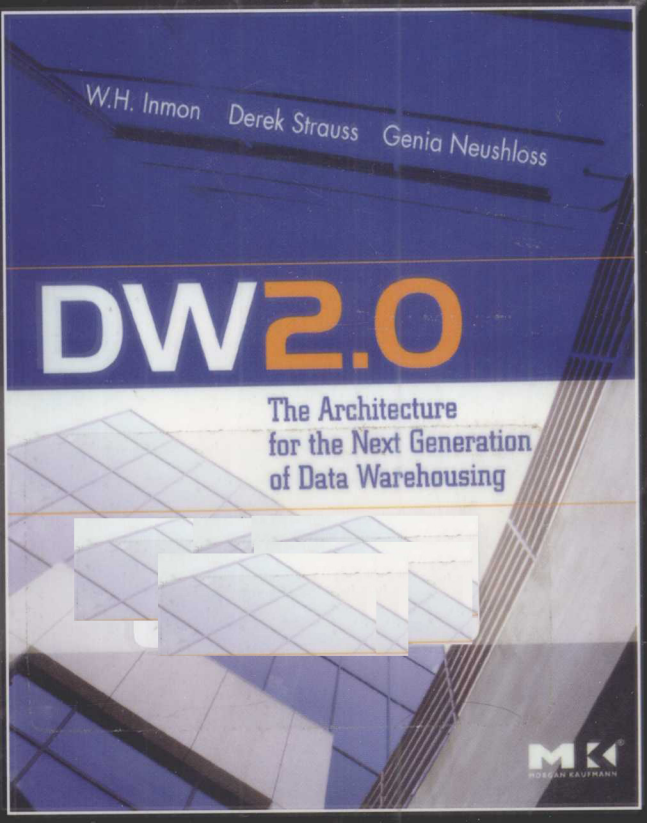
HZ BOOKS
华章教育

计 算 机 科 学 丛 书

ELSEVIER

DW2.0
下一代数据仓库的构架

(美) W. H. Inmon Derek Strauss Genia Neushloss 著 王志海 王建林 付彬 武婷婷 等译



DW2.0
The Architecture for the Next Generation of Data Warehousing

机械工业出版社
China Machine Press

DW2.0 下一代数据仓库的构架

这个行业很长时间以来就已经需要一个关于数据仓库的定义了，而DW2.0不仅仅提供了这个定义。

—— Dan Meers，弗雷德马克公司企业构架副主席

这本新的著作通过引入生命周期管理、非结构化数据和新的整合元数据的方法，使DW2.0更清晰。

—— Marvin Adams，富达投资公司共享服务主席

第二代数据仓库已经来临！在本书中，数据仓库之父向有较强信息需求的公司展示了一些技术和构架上的可能性，是数据仓库和商业智能领域的又一部经典著作。

本书包含了DW2.0详细的定义和描述，讨论了整个生命周期各个环节的具体工作，从业务需求的视角引导读者全面认识下一代数据仓库系统的构架。

本书特色

- 对租赁、技术投资、遗留系统的处理等做出正确的决策提供了具体的信息。
- 充分地解释了在数据仓库环境中非结构化数据的整合。
- 彻底地讨论了DW2.0的所有相关问题，包括非结构化数据、业务元数据、统计处理和探索处理、安全、粒度和系统性能。
- 对从DW1.0顺利迁移到DW2.0提供了专家级建议。

作者简介

W. H. Inmon 数据仓库之父。他一直致力于数据库和数据仓库技术方面的研究，在数据管理和数据仓库技术方面以及数据处理的管理方面撰写了49本著作，发表过1000多篇学术论文。他创建了世界上第一个ETL软件公司，最新成立的一个公司是Forest Rim Technology公司，该公司致力于非结构化数据的存取并将其整合到结构化环境中。

Derek Strauss Gavroshe公司的创始人、CEO和首席顾问。他拥有28年IT界从业经验和22年信息资源管理及商业智能/数据仓库领域的从业经验。

Genia Neushloss Gavroshe公司的联合创始人和首席顾问。30多年来，她在保险业、金融业、制造业、采矿业及电信业都拥有相当深厚的管理及技术经验。

客服热线：(010) 88378991, 88361066
购书热线：(010) 68326294, 88379649, 68995259
投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com

华章网站 http://www.hzbook.com

网上购书：www.china-pub.com

封面设计·锡 彬

上架指导：计算机/数据库/数据仓库

ISBN 978-7-111-28826-8

定价：45.00元

计 算 机 科 学 丛 书

DW2.0
下一代数据仓库的构架

(美) W. H. Inmon Derek Strauss Genia Neushloss 著 王志海 王建林 付彬 武婷婷 等译

DW2.0
The Architecture for the Next Generation of Data Warehousing

机械工业出版社
China Machine Press

本书是数据仓库和商业智能领域的又一部经典著作，讲述了整个生命周期各个环节的具体工作，从业务需求的视角，引导读者全面认识下一代数据仓库系统的构架。本书包含了DW2.0详细的定义和描述，此外，书中对数据仓库的结构、内容及其前景进行了介绍。

本书主要面向数据仓库的业务分析人员、信息构架师、系统开发人员、项目经理、数据仓库技术人员、数据库管理员、数据建模人员、数据管理员等。

W. H. Inmon, Derek Strauss and Genia Neushloss: DW2.0: The Architecture for the Next Generation of Data Warehousing (ISBN 978-0-12-374319-0).

ISBN: 978-981-272-335-2

本书版权登记号：图字：01-2009-2371

图书在版编目（CIP）数据

DW2.0：下一代数据仓库的构架/（美）英蒙（Inmon，W. H.）等著；王志海等译. —北京：机械工业出版社，2010.3

（计算机科学丛书）

书名原文：DW2.0：The Architecture for the Next Generation of Data Warehousing

ISBN 978-7-111-28826-8

Ⅰ. D…　Ⅱ. ①英… ②王…　Ⅲ. 数据库系统　Ⅳ. TP311.13

中国版本图书馆 CIP 数据核字（2009）第 228130 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）

责任编辑：迟振春

北京诚信伟业印刷有限公司印刷

2010 年 3 月第 1 版第 1 次印刷

184mm × 260mm · 14.5 印张

标准书号：ISBN 978-7-111-28826-8

定价：45.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991；88361066

购书热线：（010）68326294；88379649；68995259

投稿热线：（010）88379604

读者信箱：hzjsj@hzbook.com

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www. hzbook. com
电子邮件：hzjsj@ hzbook. com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章教育
华章科技图书出版中心

译者序

在过去二十年中，数据仓库的概念一直在逐步进化，DW2.0是对数据仓库概念最新的理解和描述。自从本书作者Bill Inmon首次给出数据仓库定义之后，该定义就一直被众多研究者和开发者所引用。然而，人们常常陷入什么是数据仓库或什么不是数据仓库这样的混乱或疑惑。在这种情况下，DW2.0尝试对下一代数据仓库进行全方位的定义。与术语“数据仓库”不同，DW2.0有着简明扼要和清晰可辨的含义，本书对其含义进行了详细的论述和准确的定义。

本书是数据仓库和商业智能领域的又一部经典著作，作者Bill Inmon等人在数据仓库领域享有很高的声誉，他们都长期工作在数据仓库系统开发的第一线，将自己多年的经验和感悟融入到了本书的字里行间。本书讲述了整个生命周期各个环节的具体工作，从业务需求的视角，引导读者全面认识下一代数据仓库系统的构架。本书包含了DW2.0详细的定义和描述，所有的内容被分为不同的章节，其中每一个章节都相当于该部分内容的白皮书。此外，书中对数据仓库的结构、内容及其前景进行了介绍。本书主要面向数据仓库的业务分析人员、信息构架师、系统开发人员、项目经理、数据仓库技术人员、数据库管理员、数据建模人员、数据管理员等。

本书的翻译凝结了许多人的智慧。最初，第1章由付彬翻译，第2章由李波翻译，第3章由邵金刚和李亚飞翻译，第4章由冯瑶翻译，第5章由徐闻琦翻译，第6章由王倚丹翻译，第7章与第8章由张森翻译，第9章由刘雪莲翻译，第10章由毛佳敏翻译，第11章由杨磊翻译，第12章由李志尧翻译，第13章由武婷婷翻译，第14章由郑超翻译，第15章由王鑫翻译，第16章与第17章由俞雪娇翻译，第18章由郑超翻译，第19章由邵晓康翻译，第20章、第21章和第22章由武婷婷翻译，第23章由冯瑶翻译。在此基础上，付彬和武婷婷规范了全书的术语，并进行了认真的修订。冯浩、王世强、邵鲁杰、邵进智、孙兴中、贺一航、秦逞、赵飞国、刘礼辉、王辉、张学勇、刘学军、冯岩、杨迪、黄禹钦以及王中锋等参与了本书翻译的讨论。最后，由北京交通大学王志海教授和滨州学院王建林老师审核了全书。

在翻译过程中，我们无一不被Inmon教授等人的睿智和巨大贡献所打动，秉持“形似、意似、神似”的翻译原则，尽最大的努力，希望奉献给广大读者一部真实反映原著风貌的科技书籍。

当然，要译好一本经典著作并不是一件容易的事情，我们的水平还很欠缺，错误之处还望广大读者批评指正。

译　者

2010年1月

前　言

数据仓库已经问世二十多年了，它已成为信息技术基础设施的基本组成部分。数据仓库的出现最初是为了满足对信息而不是对数据的企业需求。数据仓库是一个能够为企业提供整合的、粒度的、历史的数据的结构。

然而，数据仓库存在一个问题，即当前对数据仓库还存在多种不同的解释和实现方式。例如，有联合数据仓库、主动数据仓库、星状模式数据仓库、数据集市数据仓库等。实际上，有多少软硬件供应商，就有多少对数据仓库的诠释和实现方式。

还有一个问题就是，对什么样的结构才是数据仓库适合的，也存在着多种不同的解释和实现方式。而且，每一种实现在构架上都与其他的实现有很大区别。如果走进一个房间，里面联合数据仓库的支持者正在与主动数据仓库的支持者交谈，你也许会听到一些相同的词语，但这些词代表的意思却大相径庭。即使使用相同的词语，你听到的可能也不是有意义的交流。当两个不同背景的人交谈时，即使使用相同的词语，也不能保证他们彼此能够相互理解。

于是，今天的第一代数据仓库就处于这种情况下。

在陷入什么是数据仓库或什么不是数据仓库这样的混乱或疑惑的情况下，出现了DW2.0。DW2.0是对下一代数据仓库的定义。与术语“数据仓库”不同，DW2.0有着简明扼要和清晰可辨的含义。本书对其含义进行了论述和定义。

DW2.0中有很多重要的构架上的特征。这些构架特征代表了DW2.0相对于第一代数据仓库在技术和构架上的进步。在本书中，我们讨论了DW2.0的如下几种重要特性：

- 认识到数据仓库中数据的生命周期。第一代数据仓库仅仅将数据放于磁盘存储器（称之为仓库）中。事实上，数据一旦被置于数据仓库，它就有了自己的生命周期。进入数据仓库后，数据开始老化，数据被访问的可能性也逐渐降低。而数据访问的可能性降低对选择适当的数据管理技术有着深远的含义。另一种现象是，随着数据老化，数据容量会不断增加，并且大多数情况下这种增加是显著的。想要处理访问可能性不断降低的大量数据，就需要一种特定的设计，以免数据仓库的花费巨大，以至于不能有效地使用数据仓库。
- 当既包含结构化数据又包含非结构化数据时，数据仓库是最有效的方法。典型的第一代数据仓库完全由面向事务的结构化数据组成，这些数据仓库提供了大量有用的信息。然而，现代数据仓库应该同时包含结构化数据和非结构化数据。非结构化数据是一些文本数据，包括医疗记录、合同、电子邮件、电子表格以及很多其他的文档。非结构化数据中存在着大量的信息，但如何获取这些信息却着实是一个挑战。对创建同时包括结构化数据和非结构化数据的数据仓库都有哪些要求的具体描述是DW2.0中的一个重要部分。
- 由于多种原因，元数据并没有成为第一代数据仓库的重要组成部分。而在定义第二代数据仓库时，元数据的重要性和作用开始得到认可。在DW2.0中，问题并不是对于元数据的需求。元数据存在于数据库管理系统目录中，存在于业务对象领域中，

存在于 ETL 数据预处理工具中，等等。我们需要的是企业元数据，是从企业级视角理解元数据，需要调节元数据的所有来源并将它们放置在一个能使它们协调工作的环境中。除此之外，在 DW2.0 环境中还需要技术元数据和业务元数据的支持。

- 数据仓库最终建立在一种技术基础之上。数据仓库是围绕业务需求展开的，这通常会反映在数据模型上。随着时间的推移，企业的业务需求会发生变化，但数据仓库的技术基础却不能很容易地改变。这样，就出现了一个问题，即业务需求持续变化，而技术基础却不变。企业中这种不断变化的业务环境与相对稳定的技术环境之间的矛盾会在机构内形成很紧张的局势。在本书的相关部分中，集中讨论了两种解决方案，用于处理数据仓库中这种变化的业务需求和不变的技术基础之间的难题。一种解决方案是采用诸如 Kalido 这样的软件，其为数据仓库提供了一种有延展性的技术基础。另一种解决方案是在数据库定义时，通过设计来分离静态数据和临时数据。这两种方案对数据仓库的技术基础随着业务需求的改变而改变来说有很好的效果。

另外，书中还讨论了其他一些重要的话题。其中一些包括：

- DW2.0 数据仓库基础设施的在线更新。
- ODS 适用于哪里？
- 针对 DW2.0 数据仓库的研究处理过程和统计分析。
- DW2.0 数据仓库环境下的归档处理。
- DW2.0 数据仓库环境下的近线处理。
- 数据集市及 DW2.0。
- 数据仓库中的粒度数据和数据容量。
- 方法论及开发方式。
- DW2.0 的数据模型。

本书的一个重要特色是运用示意图来从整体上描绘 DW2.0 的环境。示意图是经过多次咨询、研讨才确定的，它代表了 DW2.0 中放置在一起的不同组件，是 DW2.0 环境的一个基本构架表现。

此外，书中对数据仓库的结构、内容及其前景进行了介绍。本书适用于业务分析人员、信息构架师、系统开发人员、项目经理、数据仓库技术人员、数据库管理员、数据建模人员、数据管理员等。

关于作者

W. H. Inmon：数据仓库之父。他已编写了 49 本著作，并被译成 9 种语言。Bill 创建了世界上第一个 ETL 软件公司。他在大多数主要的行业期刊上发表了 1000 多篇论文。

除南极洲之外，Bill 在各大洲都组织过研讨会并在各种会议上发言。他拥有九项软件专利。他最新成立的一个公司是 Forest Rim Technology 公司，该公司致力于非结构化数据的存取并将其整合到结构化环境中。每月有超过 1 000 000 人访问 Bill 的网站：inmoncif. com。他在 b-eye-network. com 上的每周通讯已经在业界被广泛阅读，每周有 75 000 个订阅者。

Derek Strauss：Gavroshe 公司的创始人、CEO 和首席顾问。他拥有 28 年 IT 界从业经验和 22 年信息资源管理及商业智能/数据仓库领域的从业经验。

Derek 发起并管理了许多企业项目，他倡导运用商业智能、数据仓库来改善数据质量。Bill Inmon 的 CIF（Corporate Information Factory）理论及 John Zachman 的 EAF（Enterprise Architecture Framework）理论是 Derek 的工作的基石。Derek 同时也是一名专家研讨会主持人，他曾多次在国内及国际的数据仓库会议中演讲。另外，他还是 DW2. 0 认证的构架师和培训师。

Genia Neushloss：Gavroshe 公司的联合创始人和首席顾问。30 多年来，她在保险业、金融业、制造业、采矿业及电信业都拥有相当深厚的管理及技术经验。

Genia 曾举办 JAD/JRP 和系统再造培训课程，是系统再造方法集的编码开发者之一。她拥有 22 年规划、分析、设计和构建数据仓库的专业经验。Genia 多次在欧洲、美国和非洲等与观众见面。另外，她也是 DW2. 0 认证的构架师和培训师。

目 录

第 1 章　数据仓库简史及第一代数据仓库

起初，人们仅用一些简单的机制来保存数据。例如，串口卡片、纸带、容量很小的磁芯存储器等。那时，存储器非常昂贵并且容量相当有限。

然而随着磁带的发明和使用，一个崭新的时代也随之来临。使用磁带能够廉价地保存海量数据。并且，磁带对数据的记录格式也没有太大的限制。另外，在磁带中数据不仅可以写入还可以重新写入。因此，与早先的存储方法相比，磁带的使用代表了一个巨大的飞跃。

然而，磁带并不是完美的。由于在磁带中是顺序地访问数据，这样为了访问其中 1% 的数据，常常可能需要物理地访问并读取 100% 的数据。另外对于写数据来说，磁带并不是最稳定的介质。磁带上的氧化物脱落或被划掉，都能导致其无法使用。

磁盘存储代表着数据存储的另一个飞跃。使用磁盘存储，可以直接访问数据，也可以重写。另外，还可以一起访问多个数据。总之，磁盘存储有着各种各样的优点。

1.1　数据库管理系统

磁盘存储产生不久，就随之产生了一种称为“DBMS”（数据库管理系统）的软件。DBMS 软件的产生是为了管理磁盘存储。磁盘存储的管理活动包括：

- 确定数据的合适位置。
- 解决当两个或多个数据单元被映射到同一个物理位置时产生的冲突。
- 允许数据被删除。
- 当无法将一条数据记录存储到一个容量有限的物理空间中时，负责为其寻找合适的物理位置。
- 其他。
- 在磁盘存储的这些优点中，数据的快速定位能力无疑是其中最重要的一个，而正是由 DBMS 完成这一重要的任务。

1.2　在线应用

一旦利用磁盘存储和 DBMS 使数据能够被直接访问后，就很快出现了所谓的在线应用。在线应用使用计算机来实现对数据的快速一致的访问。目前，已有多种商业的在线处理应用，包括 ATM（自动柜员机）、银行出纳处理、投诉处理、航空订票处理、制造控制处理、零售网点的销售处理等。简而言之，在线系统的出现使得各机构进入了能满足顾客日常需求的 20 世纪。在线应用开始变得强大并且普及起来，并且很快成长为交叉应用。

图 1-1 解释了这种信息系统的早期演化。

实际上，在线应用非常受欢迎，增长得很迅速，以至于在短期内就迅速出现了大量的应用。但是这些应用也带来了终端用户的抱怨——“我知道我想要的数据是在某个地方，

只要我能找到它。”这是个实际的情况，公司拥有了一大堆数据，但是查找数据却完全是另外一回事。并且，就算你能找出来，也不能保证你所找到的数据就是正确的。公司的数据正在激增，以至于在任何一个时间点用户都无法保证他们所获得的数据的正确性和完整性。

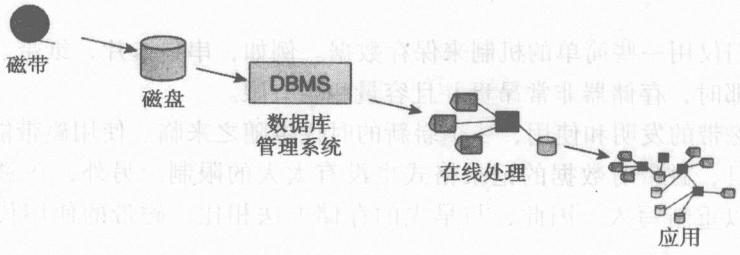


图 1-1 信息系统的早期演化

1.3 个人电脑和 4GL 技术

为了平息终端用户对访问数据的抱怨，两种新的技术应运而生——个人电脑技术和 4GL 技术。

个人电脑技术使得任何人都可以把他/她自己的电脑带进公司，并可以随意地做他/她自己的处理。出现了像电子表格（spreadsheet）这样的个人电脑软件。另外，个人电脑的拥有者可以将他/她的数据存储在自己的电脑上，这样就不再需要集中式的 IT 部门，结果就是——如果用户因为我们不让他们得到自己想要的数据而愤怒，那就给他们好了。

大约在同一时间，另一种技术也出现了，称为 4GL——第 4 代技术。4GL 蕴涵的思想是使编程和系统开发简单到任何人都可以做。这样一来，终端用户就可以摆脱必须从 IT 部门来获取企业数据的束缚。

介于个人电脑技术与 4GL 技术之间的观点是释放终端用户，这样终端用户就可以将命运掌握在自己手中。我们需要给终端用户访问其所需数据的自由，来满足他们对数据的渴望。

个人电脑技术和 4GL 技术很快就在企业中得到应用。

然而，一些没有预料到的事情在这个过程中发生了。当终端用户可以自由地访问数据时，他们发现，除了需要访问这些数据外，想要做出好的决策还有更多事要做。终端用户还发现，即使数据可以被访问，也会存在下列问题：

- 如果数据是不准确的，则没有比这更糟糕的事情了，因为不准确的数据会有很大的误导性。
- 不完整的数据的用处并不是很大。
- 不及时的数据不太符合人们的需要。
- 当同一数据出现多个版本时，依赖于其错误的值会导致糟糕的决定。
- 没有文档的数据的价值值得怀疑。

也只有在终端用户可以访问数据后，他们才能发现数据的所有潜在问题。

1.4 蜘蛛网环境

通常的结果就是一个非常大的混乱，这种混乱有时候可以形象地称为“蜘蛛网”环

境。之所以称为蜘蛛网环境，是因为有如此多的线路通向如此多的地方，这让我们想到了蜘蛛网。

图 1-2 描述了在一个典型的企业 IT 环境中蜘蛛网环境的演变。

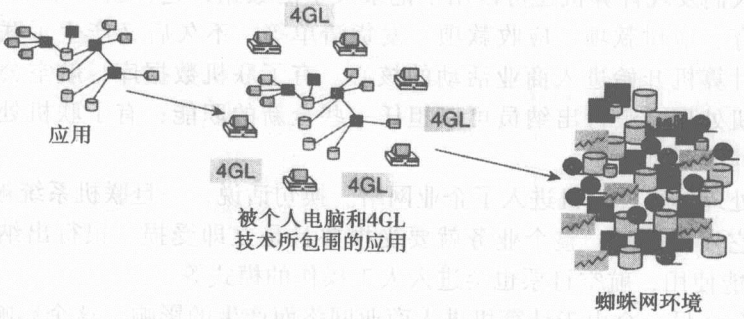


图 1-2 在一个典型的企业 IT 环境中蜘蛛网环境的演变

在许多企业环境中，蜘蛛网环境已经发展到了不可想象的复杂程度。为了证实它的复杂度，思考一下如图 1-3 所示的一个企业蜘蛛网环境的真实图表。

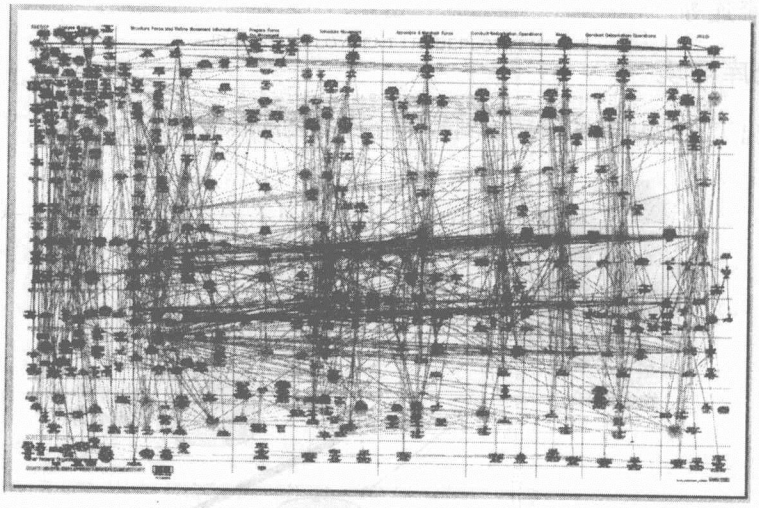


图 1-3 一个真实的蜘蛛网环境

我们看着这个图就觉得恐惧，想一想那些必须要处理如此的环境并试图用它来做一个好的企业级决定的可怜的人吧。令人惊奇的是，任何人都可以把任何事做完，不过很少人能做出好的、及时的决定。

事实上，在目前系统构架备受关注的情况下，蜘蛛网环境对企业来说是一个死胡同，想要使蜘蛛网环境工作是没有希望的事情。

终端用户、IT 专业人员和管理人员的沮丧导致了另一种不同的信息系统构架的发展，这就是以数据仓库为中心的构架。

1.5 企业角度的演化

上述过程是从技术角度出发描述的，还有一个不同的角度——企业角度。从一个企业

人员的角度出发，计算机的发展开始于重复性工作的简单自动化。与人相比，计算机能够以更快的速度、更高的准确率来处理更多的数据。例如，工资单的产生、发票的生成、正在生成的支付过程等工作都是计算机最初进入企业生活的典型应用。

不久后，人们发现计算机还可以用于记录大量的数据，这样就产生了“主文件”。主文件记录了库存、应付款项、应收款项、发货清单等。不久后又产生了联机数据库，利用联机数据库计算机开始进入商业活动的核心。有了联机数据库，航空公司的职员得以解放；有了联机处理，银行出纳员可以担任一些全新的职能；有了联机处理，保险理赔处理比以往任何时候都快。

正是联机处理使得计算机进入了企业网络。换句话说，一旦联机系统被企业人员所应用，那么如果它发生故障，整个业务就要受损并且是立即受损。银行出纳员不能做他的工作，ATM 不能使用，航空订票也会进入人工操作的模式等。

当前，还存在另一个由于计算机进入商业网络而产生的影响，这个影响关系着商业的管理、战略以及决策等方面，即当前企业决策的形成是基于在企业的动静脉等各种网络系统上的数据的。

因此，正在描述的发展过程很难说是一个以技术为中心的过程，它还伴随着一些来自企业的影响和牵连等。

1.6 数据仓库环境

图 1-4 给出了企业从蜘蛛网环境到数据仓库环境的转变。

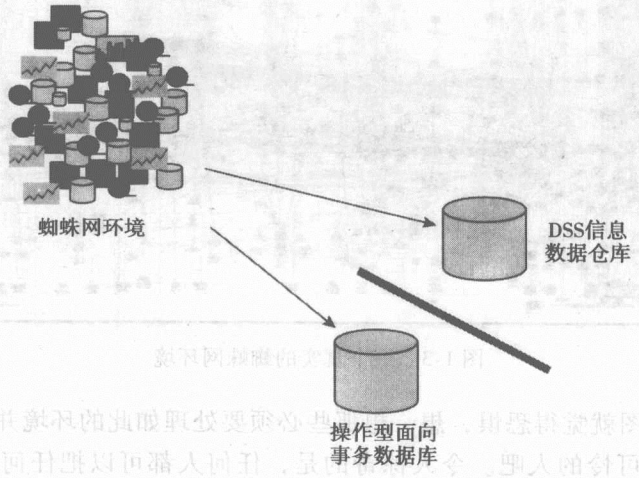


图 1-4 从蜘蛛网环境到数据仓库环境的转变

数据仓库代表了 IT 专业人员思维的重大变化。在数据仓库出现之前，人们认为数据库应该是一种能够满足所有数据需求的东西。但是随着数据仓库的出现，对多种不同种类数据库的需求变得明朗起来。

1.7 什么是数据仓库

数据仓库是信息处理的一个基础。它被定义为：

- 面向对象的。

- 整合的。
- 永久的。
- 随时间变化的。
- 一个支持管理决策的数据的集合。

这个关于数据仓库的定义从一开始就被人们所接受。

数据仓库包含了整合后的粒状历史数据。如果还有关于数据仓库的奥秘，那就是它包含的数据既是整合的又是粒状的。数据的整合使得企业对数据有一个真正的企业范围级的观察。这样，如果数据是从单一的定义良好的数据源得到的，那么就可以从整体而不是局部地观察数据来进行数据分析，显然绝大多数数据仓库不满足这一点。因此，使用数据仓库数据来观察整个企业的能力是数据仓库的首要优势。另外，数据的粒度——细分的第一层——使得数据十分灵活。由于数据是粒状的，它就可以被一组人员以一种方式考察，而被另外一组人员以另外一种方式考察。粒状数据意味着这仍旧是一组数据——关于真实状况的单一版本。财务人员可以以一种方式观察数据，市场人员可以以另一种方式观察同样的数据，会计人员还可以再用一种方式观察。如果得出了不同的结论，还可以返回那个关于真实状况的单一版本来解决这些不一致。

数据仓库的另一个优点是它是一个历史数据的集合。数据仓库是存放有价值的数年前的数据的好地方。

正是由于这些和更多的原因，数据仓库的概念才从一个被当时数据库理论家嘲笑的对象成长为今天企业中的传统观点。

然而，尽管数据仓库有着种种优势，但它也并非没有带来一定程度的痛苦。

1.8 整合数据——一个痛苦的经历

企业所感受到的第一个（也是最紧迫的）痛苦就是整合数据的需要。如果要建立一个数据仓库，就必须整合数据。可问题是许多公司都有许多遗留系统，各种各样的目的和意图使得难以处理它们。人们实在不情愿对他们的旧遗留系统做任何改变，但是建立一个数据仓库又要求他们不得不这么做。

因此，建立数据仓库的第一个障碍就是“弄脏你的手”，即返回那些旧的遗留系统，看看你都有哪些数据，然后弄明白如何将这些面向应用的数据转化成企业数据。

这种转化绝非容易，并且在某些情况下几乎不可能。但是整合数据带来的价值值得我们去承受转化未整合、面向应用的数据过程中的痛苦。

1.9 数据的量

数据仓库所面临的第二个痛苦是处理数据仓库产生的大量的数据。大多数 IT 专业人员以前从来没有处理过伴随数据仓库产生的如此大量的数据。在应用系统环境下，尽早地丢弃较老的数据是个好的实践方法。在操作型应用环境下并不需要旧的数据，因为它会使系统慢下来。旧的数据阻塞了数据流通的要道。因此，任何一个好的系统程序员会告诉你如果想要使系统变得高效，就必须丢弃旧的数据。

然而，在旧的数据中仍然存在巨大的价值。对许多分析来说，旧数据是极其有用的，有时甚至是不可或缺的。因此，有一个合适的地方（例如数据仓库）来存储旧数据对于

数据分析而言太有用了。

1.10 一种不同的开发方法

数据仓库所面临的第三个痛苦是构建数据仓库的方法，这并不是能轻易完成的。全世界的开发者习惯于先收集需求然后构建一个系统。这种历史悠久的方法已经在开发者构建运行系统的时候被反复地灌输到他们的头脑当中了。但是数据仓库的构建却非常不同，它是迭代地被构建的，每次前进一步，先构建一部分然后再构建另一部分，如此等等。几乎在每次开发实例中，这种方法都作为一种策略来应对那些试图使用“激进”的方法一次构建好整个数据仓库所带来的灾难。

关于构建数据仓库不应该使用激进方法有许多原因。第一个原因就是数据仓库项目一般都比较大。有一句古话说得好：“你怎样才能吃掉一只大象？如果尝试一口气吃掉，那你就会被噎住。相反，吃大象的方法应该是每次吃一点。”当我们构建数据仓库的时候，这个逻辑是再正确不过的了。

关于构建数据仓库应每次构造一部分还有一个好的原因。这个原因就是最初构建数据仓库的时候，对于它的需求并不总是明确的。这也是因为数据仓库的终端用户并不确切地知道他们想要什么。终端用户以一种探索的模式进行操作。他们怀着这样的态度——“只有当我看见可能发生的事情时，我才能告诉你我真正想要的是什么。”而正是构建数据仓库的第一次迭代活动开阔了终端用户的思路，引导用户去考虑可能发生的事情会是什么。也只有在看见数据仓库后，用户对它的需求才能变得明确。

问题是：传统的系统开发者之前从未以这种方式构建这样一个系统。而当开发者仅仅把它当作是另一个操作型应用系统来开发时，会带来数据仓库构建过程中最大的失败。

1.11 演变到 DW2.0 环境

本章已经描述了一个从最早的系统发展到 DW2.0 环境的演变过程。从构架演变的角度出发，回顾并考察使得这一演变过程成形的推动因素是非常有趣的。事实上，有许多因素推动了这一信息构架演变的形成，并达到其最高点——DW2.0。

其中一些演变过程的推动因素为：

- 对于更多不同技术的使用需求：当比较一个最初的系统和 DW2.0 的系统时，可以发现 DW2.0 在系统及其与终端用户的交互方面已经有了显著提升。而在不久之前计算机系统以穿孔卡片的形式输出的时候，这几乎是不可想象的事情。终端用户的输出作为一个微小的信息点被掩埋在十六进制堆中。事实上，只要输出还是以这种非常原始的形式出现，计算机就不算是很有效的。
- 联机处理：只要对数据的访问被限制在一段非常短的时间，商业人士就可以利用电脑来做很多事情。但是联机处理一旦成为可能，商业活动就会使交互使用日常商业活动中的信息成为可能。有了联机处理，预定系统、银行出纳处理、ATM 处理、联机目录管理以及其他一大堆的关于计算机的重要应用就会成为现实。
- 对于整合的企业数据的渴望：在仍存在大量应用的时候，这种来自办公室的渴望被扑灭了。但是不久以后，人们就发现有一些重要的东西被遗漏了，而被遗漏的正是企业信息。企业信息无法通过将若干微小的应用加在一起而获得，相反数据

必须被改造为整合的能为企业所理解的信息。但是一旦企业数据成为现实，对于处理的所有新的看法将被开启。

- 对于混合地包含非结构化的文本数据的需要：多年以来，决策都是仅仅在结构化的记录数据这一基础上做出的。虽然结构化的记录信息确实非常重要，但在企业环境中仍然存在其他的信息形式。有大量的信息以文本的、非结构化的形式存在。不幸的是，抽取出这些文本的信息并不是容易的。但幸运的是，文本 ETL（抽取/转换/装载）出现了，并为各种组织提供了获取作为制定决策基础的文本信息的关键方法。
- 容量：如果技术世界停止了创新，一个像 DW2.0 这样复杂的世界就完全不可能出现了。但是技术的容量、技术工作的速度，以及使不同形式的技术可以互相联系起来的能力合起来创造了一个这样的技术氛围，其中容量是一个常见的制约。可以想象这样一个世界：所有的存储全部保存在磁带上（就像不久前一样），那么，绝大多数现在被认为是理所应当的处理类型完全是不可能产生的。
- 经济效应：除了容量的增长，技术的经济效应对客户也是非常有利的。如果顾客还必须像十年前那样为技术埋单，那么从金融学的角度看，DW2.0 的数据仓库就完全地偏离了轨道。多亏了摩尔定律，很多年来技术的单位成本已经缩减了，最终达到客户层的可支付能力。

这些就是过去几十年来技术世界的一些进化推动因素，并且这些推动因素促进了构架的演变，其集中体现就是 DW2.0。

1.12 数据仓库的商业影响

数据仓库对于商业的影响是非常大的。一些直接受到数据仓库出现的影响的领域包括：

- 航空业的常旅客计划：常旅客计划拥有的最有价值的一项技术就是它们的中心数据仓库。
- 信用卡欺诈分析：每一个客户都在其过去的消费行为基础上产生一些消费记录。这些记录是从数据仓库中生成的。当一个客户试图进行一个超过其记录范围的购买时，信用卡公司就会检查是否将要发生信用卡的欺诈性使用。
- 详细目录管理：数据仓库保存了详尽的存货记录、注意趋势及机会机遇。通过理解——在一定的细节层次上——一个组织所管理的货物的消费模式，公司可以同时了解供给过剩和供给不足的情况。
- 客户记录：那些想要“更好地了解他们的客户”的组织跟踪保留了他们的客户所展示的购买模式和注意力模式。这些详细的信息都被存储在数据仓库中。

数据仓库还通过许多其他的途径影响商业活动。简而言之，数据仓库成为了企业的存储器。没有数据仓库时，至多也就是有一个短期的企业存储器，而有了数据仓库就等于有了一个长期的、详细的企业存储器，并且可对该存储器以不同的方式加以利用。

1.13 数据仓库环境的各种组件

数据仓库环境中有着各种组件。起初，这些组件并没有被广泛认可。但是不久以后这

些数据仓库的基本组件就被熟知了。

图 1-5 展示了从一个早期的独立的数据仓库到一个完整的数据仓库构架的发展过程。

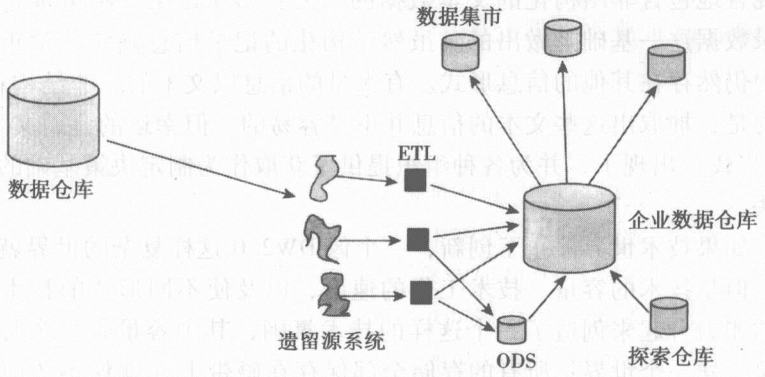


图 1-5 数据仓库很快就演变为一个被称为企业信息工厂的成熟构架

在图 1-5 中给出的完整的构架包含了一些常见的组件，这些组件将在以后的章节中讨论。

1.13.1 ETL——抽取/转换/装载

ETL 技术使得数据可以从遗留系统环境中获得并被转换成企业数据。ETL 的组件执行许多功能，例如：

- 数据的逻辑转换。
- 域的验证。
- 从一个 DBMS 到另一个的转换。
- 当需要时，默认值的生成。
- 数据的总结。
- 对数据键添加时间值。
- 重构数据键。
- 记录的合并。
- 额外或冗余数据的删除。

ETL 的本质是使数据进入 ETL 迷宫时是应用数据，而出来时则变成了企业数据。

1.13.2 ODS——操作数据存储

ODS 是在联机交易处理（OLTP）响应时间内完成整合数据的联机升级的地方。ODS 是一个复杂的环境，在其中应用数据被转换（通常使用 ETL）成整合的形式。一旦被放进 ODS，数据就可以得到高性能的处理，包括升级处理。在一定程度上，ODS 使传统的数据仓库避开了应用数据以及在实时模式的升级过程中事务完整性和数据完整处理的总开销。

1.13.3 数据集市

数据集市是终端用户可以直接访问和控制所分析数据的地方。数据集市是根据一组部

门用户对数据应该以何种方式被看到的一般期望形成的。财务部有它自己的数据集市，市场部有一个数据集市，销售部也有一个数据集市，等等。每一个数据集市的数据来源都是数据仓库。数据集市通常是用不同的技术而不是不同的数据仓库来实现的。每一个数据集市包含的数据通常比数据仓库少得多，它通常也包含大量的汇总数据以及聚合数据。

1.13.4 探索仓库

探索仓库向想要对数据进行发掘处理的终端用户提供了相应的功能设备。许多统计分析就是在探索仓库中完成的。许多在探索仓库中进行的处理都属于不同类型的启发探索。大多数探索仓库都是基于一个项目保存数据，一旦项目完成了，探索仓库也就可以不用了。探索仓库承担了重要的统计分析的处理要求，这样就使传统的数据仓库避开了由于使用探索仓库做非常繁重的统计而引起的性能缺失。

这种构架已经被通称为企业信息工厂。

这样，简单的数据仓库的概念已经从一个用于存放整合的、粒状的、历史的数据的地方演变成为一个完整成熟的构架。

1.14 数据仓库的演变——从企业的角度

在计算的最初期，终端用户以一种非常原始的方式从计算机得到输出。在那个时候，终端用户要读打在卡片上的孔，并且要读用数千页隐晦的代码才能保存一点信息的十六进制堆。不久后报表变得规范，因为最早期的对终端用户的接口形式确实是太原始了。

不久后，终端用户变得复杂起来。终端用户得到的能力越大，他们能够想象到的能力也就越大。在报表出现后，联机信息也几乎同时出现并可用了。

并且，在联机交易处理后，终端用户又想要整合的企业数据，通过它可以将大量的数据整合成一个聚合的整体。之后，终端用户又想要历史数据。

在此过程中也同时贯穿着构架和技术的演变。而正是通过第一代数据仓库，终端用户才到达了分析能力的终极。

换一种说法，如果没有第一代数据仓库，终端用户对信息仅会有局部的、不完整的需求。终端用户对企业信息的渴望是第一代数据仓库发展背后的最大推动力。

1.15 关于数据仓库的其他观念

然而，还存在着其他的力量在改变着关于数据仓库是什么的观念。一些计算机经销商已经认识到数据仓库是个非常吸引人的东西，所以他们“更改”了数据仓库的概念以满足其需要，即使他们从来不用数据仓库做那些在广告中宣称的事情。

一些经销商和咨询师更改数据仓库的方式如图 1-6 所示。

图 1-6 展示了现今数据仓库的一些变体，特别是：

- “主动”数据仓库。
- “联合”数据仓库。
- “星状”数据仓库（有着规范的维度）。
- “数据集市”数据仓库。

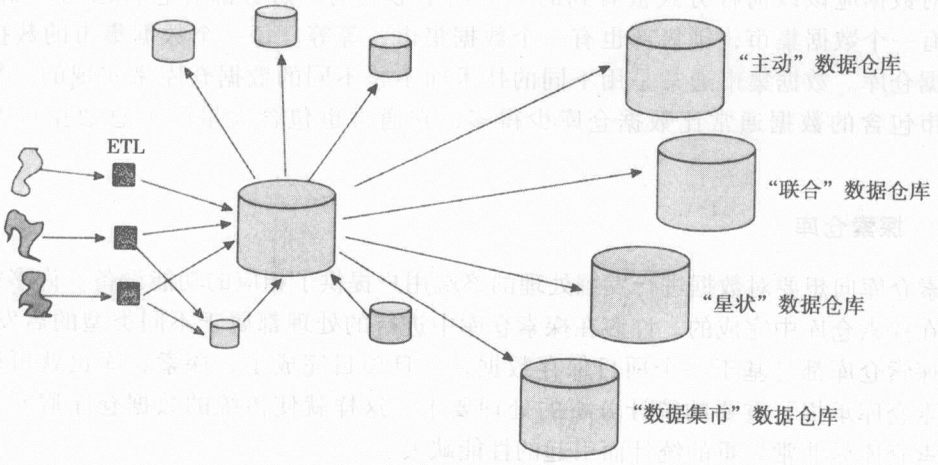


图 1-6 不同的数据仓库很快就开始出现

虽然这些更改过的数据仓库概念的每一个都和真正的数据仓库有一些相似，但在数据仓库和它的这些变体间仍存在一些较大的区别，并且每种更改后的变体都有一些较大的缺陷。

1.16 主动数据仓库

主动数据仓库是指在其中可以完成联机处理和升级。高性能的事务处理是主动数据仓库的一个特征。

主动数据仓库的一些缺陷包括：

- 维护数据和事务的完整性的困难：当一个事务没有被正确地执行而需要放弃时，在需要找出或毁坏或修正的数据时会遇到问题。尽管这些事都能完成，但通常非常复杂并需要相当多的资源。
- 容量：为了保证良好的联机响应时间，必须有足够的容量来确保在高峰时期处理时间有足够可用的资源。虽然这点肯定能满足，但结果通常是有大块的容量在长时间内没有被利用，导致操作成本非常高。
- 统计处理：繁重的统计处理与标准的数据仓库处理的系统资源利用之间的冲突总是一个问题。不幸的是，主动数据仓库的销售商却宣称他们采用的技术能够解决这个问题。
- 成本：主动数据仓库环境是昂贵的，这有无数的理由——从等待高峰时期处理的未使用的容量到所有的细节数据都必须存储在一个数据仓库的观念，甚至对那些数据的访问的可能性早已减小等。

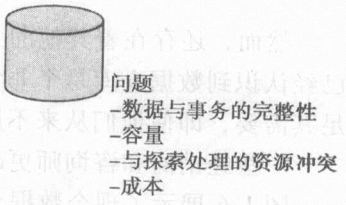


图 1-7 主动数据仓库

图 1-7 给出了主动数据仓库方法的一些缺点。

1.17 联合数据仓库方法

在联合数据仓库方法中根本没有数据仓库，因为企业都惧怕整合数据的工作。这种方法就是把那些老的遗留数据库粘合在一起虚拟地建立一个数据仓库，这些数据库的数据

可以同时被访问。

联合数据仓库方法是非常吸引人的，因为看起来它给了企业避免数据整合的选择。不管在什么地点什么时候，要完成旧的遗留系统的整合都是一项巨大且复杂的任务。除非你使用这种联合的方法，否则就不得不整合旧的数据。

但不幸的是，这种联合式方法更像是一种假象而不是一种解决方案。这种方法有许多根本性的问题，如下所示，但还不止这些：

- 低劣的性能：关于联合的数据能导致低劣的性能有许多原因。如果一个需要加入到联合式数据仓库的数据库出现故障或正在重组该怎么办？如果联合数据仓库需要的数据库正在参与 OLTP 该怎么办？是否有足够的机器周期来同时满足 OLTP 和数据库联合的需要？如果相同的查询语句被执行两次，或者如果不同的查询需要相同的数据该怎么办？每次当需要的时候，它都要访问并联合，这对于资源的使用来说是一种浪费的方法。以上还仅仅是性能为何成为联合环境的一个问题的部分例证。
- 缺乏数据整合：这种联合式方法下根本没有数据的整合。如果正在被联合的数据已经经过整合了，那好极了；但这种情况很少出现。联合式方法对数据整合一无所知。如果一个文件中有美元，另一个文件中有加元，第三个文件中有澳元（这三种货币使用相同的符号 $ ——译者注），那么在联合的过程中，这三种货币就会被加在一起。数据整合的基本问题是联合式方法中一个现实而严重的问题。
- 复杂的技术方法：即使实现了联合，也需要一个复杂的技术方法。例如，联合需要利用多个厂商的 DBMS 技术才能够完成。假如不同的数据库厂商不愿意互相合作，那么，既要基于这些不同的数据库又要整合它们就成了一个有问题而不可靠的技术，这种情况不足为奇。
- 有限的历史数据：对于联合的数据库来说，仅有的可用历史数据就是那些已经存在于联合数据仓库中的数据。几乎所有情况下，这些数据库都会为了追求性能而尽可能快地丢掉历史数据。因此，通常仅有一小部分历史数据能够进入联合数据仓库中。
- 不可重复的查询：在这种联合式的环境中，查询不具有可重复性。假设在上午 10 点的时候提交了一个对数据库 ABC 的查询并且返回一个值 $156.09。接着，在上午 10 点半的时候，一个客户进来向其账户中添加了一笔工资储蓄，将账户的值改成了 $2 971.98。在上午 10 点 45 分的时候再重复上次的联合查询。在不到一个小时之内，对相同的查询将会返回一个不同的值。
- 继承的数据粒度：联合数据仓库的用户受困于诸如在支持联合查询的应用中找到的粒度。只要粒度已经存在于支持联合查询的数据库中，那它就是客户想要的，这没有问题。然而当客户想要另一个不同层次的数据粒度——或者高也或者低——的时候就会出现一个严重的基本问题。

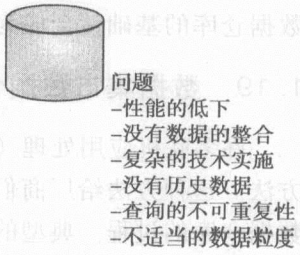


图 1-8 联合数据仓库

图 1-8 总结了关于数据仓库联合方法中存在的基本问题。

1.18 星状模式方法

数据仓库的星状模式方法要求建立事实表和维度表。使用星状模式方法能实现实际的数据仓库的许多优点。然而，这种方法也存在一些基本的问题，包括：

- 脆弱性：由一些星状模式的集合组成的数据仓库有些“脆弱”。只要需求都精确地在同一时刻开始，那就没有问题。但是一旦需求发生变化（久而久之这种变化是肯定会发生的），那么，要么对现有的星状模式做大块的改动，要么就丢弃掉它并用新的来取而代之。事实是，星状模式是针对且仅针对一组给定的需求来设计的。
- 有限的可扩展性：相似地，星状模式是很难扩展的。它们一开始是在需求的基础上设计的，并被这些需求极大地限制着。
- 一类用户：星状模式为了某一类用户的使用而进行优化。通常，仅有一组用户感到所给定的星状模式是最优的，而其他的用户都感觉它不是最优的。一个数据仓库的本质任务是使大量的不同用户满意。如果存在这样的情况，即有的用户对服务方式并不太满意，那么这样的数据仓库就不是最优的。而建立一个单一的星状模式来为所有用户服务恰恰就会造成这种情况。
- 星状模式的增殖：由于一个单一的星状模式并不能最优地满足一个大的团体用户的需求，所以通常会尝试建立多个星状模式。当建立多个星状模式后，每个星状模式都不可避免地有不同的粒度层次，并且数据整合就成了问题。这种星状模式的增殖使得寻找数据的企业级视图几乎不可能。
- 退化：为了解决不同星状模式间多粒度级的问题，每个星状模式的数据都必须使用最低一级的粒度。这样违背了首选星状模式的理论，并产生了一个典型的相关设计，这种设计让数据仓库的设计者感觉格格不入。

图 1-9 显示了数据仓库使用星状模式带来的挑战。

从长期来看，星状模式对于数据仓库来说并不是非常好的。当有大量的数据和大量的用户时，即有大量的多样性，使星状模式能够有效工作的唯一方法就是确保数据是非冗余的并且在最低级的粒度上对其建模。即使这样，当需求发生变化时，星状模式仍可能不得不被修改或替换。

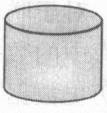


图 1-9 星状模式数据仓库

然而，在一个用户理解数据的方式几乎没有多样性的环境中，如果需求随着时间并不改变，并且如果没有太多的用户，那么星状模式作为数据仓库的基础就会成为可能。

1.19 数据集市数据仓库

许多联机应用处理（OLAP）技术的厂商都沉醉于使用数据集市作为建立数据仓库的方法。这种方法给厂商们提供了机会先销售他们的产品，而不需要经历建立一个真正的数据仓库的过程。典型的 OLAP 厂商的销售定位为——“先建立一个数据集市，然后再将它转化为一个数据仓库。”

不幸的是，在建立一个数据集市集合并将它们称为数据仓库的过程中仍存在着许多问

题。其中一些是：

- 数据的不可调和性：当管理人员问这样一个问题："上个月的收入是多少?"，会计会给出一个答案，市场人员会给出另一个，并且财政人员也会给出一个。当会计、市场人员、财政人员的答案互相不一致时，其调和是非常困难的。
- 抽取增殖：当数据集市第一次建立起来的时候，对原始环境的数据抽取的数目是可接受的，或者至少是合理的。但是当越来越多的数据集市被加进来时，越来越多的原始数据抽取也必须加进来。这样，到某一时刻，抽取原始数据的负担就会变得无法忍受。
- 变更的传递：当有多个数据集市并且必须做一些变更时，这些变更就会涉及所有数据集市。如果在一个财政数据集市中必须做一个变更，就很有可能也得在销售数据集市中再做一遍该变更，同样也会在市场数据集市中再做一遍，等等。当有多个数据集市时，必须在多个地方实施变更。此外，这些变更必须以相同的方式实施。不能以一种方式在财政数据集市中实施变更，而以另外一种方式在销售数据集市中实施变更，否则出错率就会以一种前所未有的速度迅速增加。用于确保传递的变更是正确且彻底完整的管理人员、时间、资金以及准则等已经超过了许多组织的承受范围。
- 不可扩展性：当需要建立一个新的数据集市时，不幸的是大多数情况下我们必须从头建起。目前为止仍没有一种现实可行的方法，使得建立新的数据集市时能够大量利用原来建立数据集市时所做的工作，甚至一点都用不到。

所有这些因素加起来就构成了利用数据集市作为构建数据仓库的方法的现实，企业会陷入一场维护的噩梦中。

图 1-10 描述了使用数据集市作为构建数据仓库的方法的问题。

一个有趣的情况是，建立一个数据集市然后使它成长为一个数据仓库是不可能的。数据集市的 DNA 从根本上就是不同于数据仓库的 DNA 的。建立一个数据集市然后使它成长为一个数据仓库的理论类似于宣称如果你种下了一些蒲公英，而它们长大后会是橡树。蒲公英的 DNA 和橡树的 DNA 是不同的。为了得到橡树，你必须种植橡子。种植蒲公英的种子只会得到蒲公英。实际上，橡树和蒲公英在春季开始生长的时候，这两种植物看起来都是一样的小绿芽仅仅是个巧合。

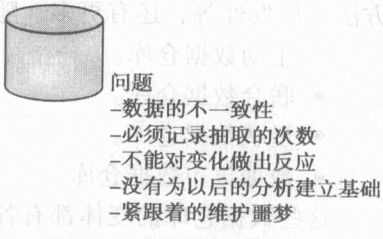


图 1-10 数据集市数据仓库

同样，数据集市和数据仓库之间存在一些相似性，但认为其中一个也是另外一个或者认为其中一个可以变为另外一个都是错误的观点。

1.20 建立一个"真正的"数据仓库

开发者在构架层次和数据仓库构建过程的开始会做一些重要的选择。主要的选择是需要构建什么类型的数据仓库——一个"真正的"数据仓库还是某一数据仓库的各种变形中的一个？这个选择意义长远，因为构建一个数据仓库所需的财力和人力成本通常是非常高的。如果开发者做了一个错误的选择，那么后来某时刻肯定得重复做许多费力的工

作。没有人喜欢浪费大量的资源，而且也很少有人能负担得起。

图 1-11 显示了经理或构架师正面临的两难困境。

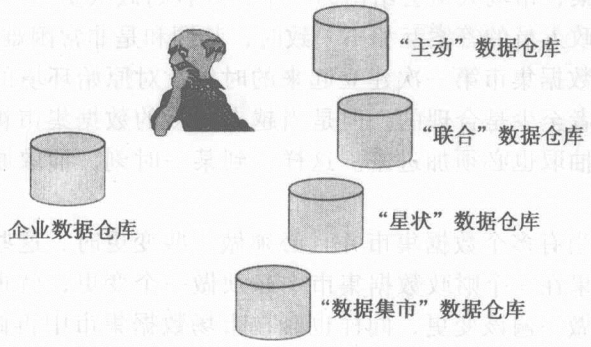


图 1-11 所有长期或短期的后果对组织非常重要

做选择时面临的问题之一是正在兜售数据仓库的厂商都非常善于游说，他们的第一目标就是说服客户去构建需要他们的产品和服务的数据仓库，而不是满足商业需求的那种必需的数据仓库。不幸的是，落入这种销售圈套可能会浪费大量的资金和时间。

1.21 总结

从用户数据仅限于通过 IT 部门中介访问到的可操作应用数据的那些令人沮丧的日子开始，数据仓库已经经过很长的一段发展时间。现在，数据仓库已经进化到可以满足终端用户对整合的、历史的、粒状的、灵活的以及准确的信息的需求了。

第一代数据仓库已经变得包含从粒状的、历史的、整合的数据仓库的原始应用中得到的训练有素的数据 ETL。随着数据仓库的流行，也出现了许多挑战——数据量、空间开发方法、启发性等，还有更多。随着数据仓库的演变的继续，一些变体也出现了：

- 主动数据仓库。
- 联合数据仓库。
- 星状数据仓库。
- 数据集市数据仓库。

这些数据仓库的变体都有各自的优点，但它们同样也都带了许多新的、明显的缺点。下一代数据仓库的时代来临了。

第2章　DW2.0简介

为了解决数据仓库构架的选择问题并清除所有的干扰，人们制定了DW2.0。DW2.0是为下一代数据仓库定义的数据仓库构架。为了理解DW2.0是怎样形成的，考虑其以下几个形成因素：

- 在第一代数据仓库中，强调了数据仓库本身的建立和增加商业价值。在第一代数据仓库的时代，获得价值是指主要获取以数字为主的事务数据，并整合它们。而今天，从企业数据中获得最大价值意味着利用所有的企业数据并从中获取价值，这意味着既包括文本的、非结构化的数据，也包括数字化的交易数据。
- 在第一代数据仓库中，没有对数据的存储介质和数据量给予太多关注。但时间已经证明数据的存储介质和数据量确实是非常重要的问题。
- 在第一代数据仓库中，人们已经认识到整合数据是一个问题。而现在，人们发现整合旧的数据是一个超乎想象的更大的问题。
- 在第一代数据仓库中，成本几乎不用考虑。而现在，数据仓库的成本则是人们的一个主要关注点。
- 在第一代数据仓库中，人们忽视了元数据。而现在，元数据和主数据的管理成为人们热议的问题。
- 在第一代数据仓库的早期，数据仓库被认为是一个新鲜事物。如今，数据仓库被认为是有竞争力地利用信息的基础。数据仓库已经变得必不可少。
- 在数据仓库发展的早期，重点仅仅是构建数据仓库。现在人们认识到，数据仓库需要随着时间的推移保持可扩展性，以便跟得上不断变化的业务需求。
- 在数据仓库发展的早期，人们认为数据仓库对统计分析可能有用。今天人们认识到，利用数据仓库进行统计分析的最有效方法是在一个被称为探索仓库的相关数据仓库结构中进行。

如今，经过几十年建立和使用这些结构的实践，我们确实已经对数据仓库更加了解。

2.1　DW2.0——一种新的范式

DW2.0是由当前的一些开明且有远见的决策支持企事业界所要求的一种新的数据仓库范式。这种新的范式关注数据的不同类型、基本结构，以及它们怎样关联起来形成一个强大的数据存储器以满足公司对信息的需求。

图2-1解释了新的DW2.0构架。该图展示了不同的数据类型、它们的基本结构，以及不同的数据类型如何关联。本书接下来将致力于介绍隐含在该图中的DW2.0下一代数据仓库构架及其微妙之处。

2.2　DW2.0——从企业的角度

DW2.0之所以能吸引企业人士有一些重要的原因。其中的一些是：

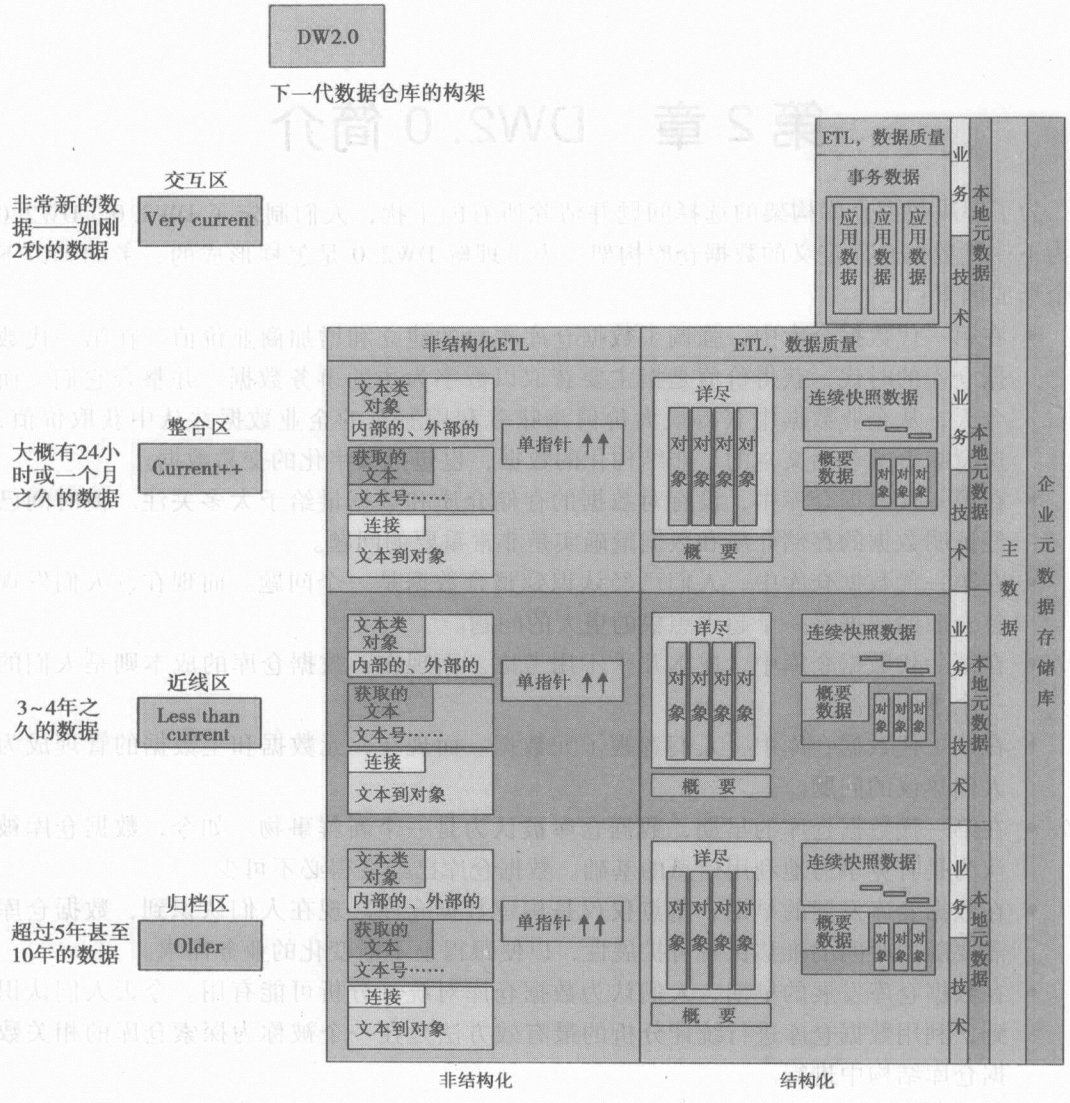


图 2-1 DW2.0 的数据结构

- 数据仓库基础设施的成本不再持续增长。在第一代数据仓库中，技术基础设施的成本是不断增长的。随着数据量的增长，基础设施的成本会以指数级增长。但是使用 DW2.0，数据仓库的这一成本会趋于平稳。
- 使用元数据将基础设施结合在一起，这意味着数据不会轻易丢失。在第一代数据仓库中，一个数据单元或一个数据类型是很容易“丢失”的。这就像纽约市立图书馆书架上的一本书摆错了位置一样，一旦摆错位置，可能需要若干年才能将其放回容易被人们找到的位置。第一代数据仓库环境下的数据也是如此。而作为 DW2.0 骨干的元数据则不会令数据轻易丢失。
- 数据访问速度快。在第一代数据仓库中，数据是堆叠在其他数据上的。堆叠的数据很快就成为访问障碍，因为需要的数据很可能“隐藏”在大量不需要的数据之

下，结果造成较低的访问性能。而在 DW2.0 环境下，数据是根据其访问概率放置的，因此它的数据访问性能比第一代数据仓库环境更为有效。

- 存档需求的关注。第一代数据仓库中很少有甚至没有存档数据，因此数据只能存储相对较短的一段时间。而在 DW2.0 环境下，数据是被存档的，这样它就能够永久保存下去，或者视需要而定。
- 数据仓库吸引大量的数据。使用第一代数据仓库，终端用户不得不忍受管理和访问大量数据带来的痛苦。而对于 DW2.0，由于数据是分段的，终端用户需要处理的数据量就会少得多。

所有这些因素都对终端用户有一定影响。数据仓库的成本显著降低，有效访问和查询数据的能力提高，数据访问速度加快，数据可保存的时间增长。简而言之，这些因素提高了企业人士使用数据的能力，使他们能够以一种比第一代数据仓库更有效的方式使用数据。

那么，DW2.0 与第一代数据仓库有哪些区别呢？事实上有很多显著的差别，最突出也是最重要的一点就是对数据仓库中的数据生命周期的认识。

2.3 数据的生命周期

在第一代数据仓库中，人们认为只需要在数据仓库建立时把数据存放在某种形式的磁盘存储器即可。但是，这仅仅是个开始——数据进入数据仓库后就开启了生命周期。另外，这也只是第一代数据仓库开发者的天真想法。

认识到数据在数据仓库中的生命周期后，DW2.0 数据仓库包括了四个数据生命周期“分区”。第一个分区是交互区。数据存入数据仓库后迅速进入交互区。随着数据的调整，数据被整合后传递到整合区。毫无疑问，整合的数据是在整合区被发现的，并且一直位于整合区，直至其访问概率下降。数据的访问概率往往会伴随着存储时间的增加而下降。通常情况下，3～4 年之后，整合区数据的访问概率会明显下降。

数据经过整合区之后可能进入以下两个分区之一。一个是近线区。在许多方面，近线区就像是整合区的延伸。近线区是可选择的，亦即数据不一定需要经过这一区。但是当数据量非常大并且数据间的访问概率差别很大时，就可以利用近线区来处理。

另一个分区则是归档区。归档区中的数据访问概率很低，数据既可以从近线区也可以从整合区进入归档区。归档区的数据通常是 5～10 年，甚至更长。

数据的生命周期在它进入 DW2.0 数据仓库后看起来是什么样的呢？图 2-2 说明了 DW2.0 的数据生命周期。

数据要么通过 ETL 从另一个应用程序导入 DW2.0 环境，要么通过嵌入在交互区中的应用程序直接导入 DW2.0 环境。交互区是数据联机更新的场所，并且在响应时间方面有着很高的性能。进入交互区的数据都是刚进入数据仓库的新数据，也许只生成了几秒钟。交互区的另一类数据被用作共享应用程序的一部分来处理，在这种情况下，数据生成时间仅有几毫秒。

作为交互区的数据的一个实例，我们来考虑一次自动柜员机（ATM）的交易。在一次自动柜员机的交易中，数据会在交易一结束就被获得。这些数据在大约不到一秒钟的时间内送入交互区。数据可以利用两种方式之一进入交互区。一种方法是在 DW2.0 数据

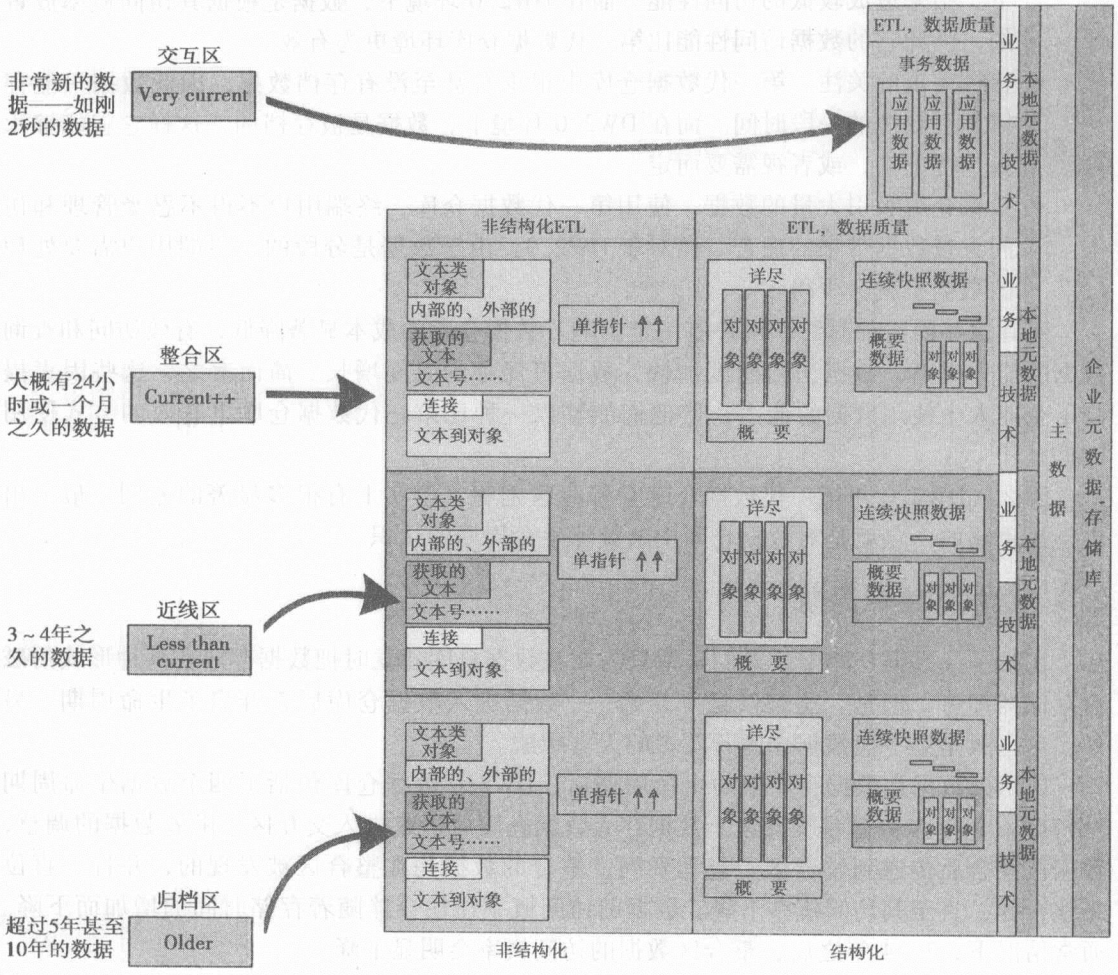


图 2-2 数据在 DW2.0 环境下有生命周期，在数据仓库中的不同阶段也有相应的数据区

仓库外部可能存在这样的应用，可以捕获被当作交易的副产品的数据。在这种情况下，应用程序执行交易，之后将数据通过 ETL 传送到数据仓库的交互区。

数据进入交互区的另一种方式是在应用程序作为 DW2.0 数据仓库的一部分的情况下，应用程序执行交易，之后立即将数据送入交互区。

区分两种方式的关键在于交互区的应用程序位于交互区的外部，还是实际位于交互区内。

无论其起源如何，交易数据必定是面向应用的，在交互区的数据最终会到达应用状态。

在有的时间点，需要将交易数据与应用数据整合在一起。这些时间点可能在数据到达交互区之后的几秒，也可能是几天或者几周之后。无论如何，在一些时间点需要整合应用数据，这些时间点通常是在数据进入整合区之后的时间。

数据通过 ETL 进入整合区。在经 ETL 进入整合区的同时，数据脱离了应用状态，并获得企业数据状态。这个任务是由 ETL 的转换代码来完成的。

一旦进入整合状态，数据与其他相似的数据聚集在一起。大量数据聚集在整合区，而且只要其一直保持较高的访问概率，就会一直处于整合状态。对于很多组织而言，这意味着数据在整合区要保留 3 ~ 5 年的时间，这取决于组织的业务及其所做的决策支持处理。

在某些情况下，整合区将有非常大的数据量和非常频繁的数据访问。这时，最好使用近线存储器作为整合数据的一个缓存。企业可以利用近线区以电子方式提供大量数据。带有近线存储器的整合数据存储器使得整个整合环境的成本更易接受。当数据的访问概率剧烈下降时，数据被放入近线存储器，而访问概率大的数据则不应存放在近线存储器内。我们认为所有存入近线存储器的数据的访问概率都已经由控制企业数据存储的分析员核实过。

DW2.0 的最后一个区是归档区。归档区保存那些以电子方式收集回来的、将来可能被使用的数据。归档区所存储的数据是由近线区或者整合区传送来的，它们的访问概率很低。在某些情况下，将数据存储在归档区是出于预防的目的，即使人们认为它的访问概率是零。

2.4 设置不同区的原因

在 DW2.0 环境下设置不同的区有多方面的原因。不同区之间区别的核心问题是，数据从一个区传递到另一个区时，数据的基本操作参数随之改变。图 2-3 展示了各个区之间基本操作上的一些差异。

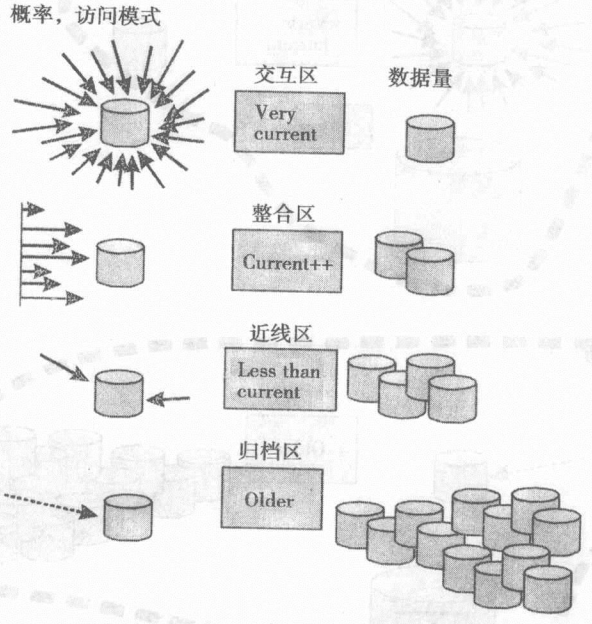


图 2-3 随着 DW2.0 中的数据经历其整个生命周期，它的访问概率和数据量显著变化

图 2-3 表明，在区与区之间，数据的访问概率和访问模式差别很大。交互区的数据被频繁访问，并且其访问模式是随机访问。整合区数据的被访问概率也很高，但通常是顺序、成串的访问。近线区的数据访问概率相对较低，并且在访问时是随机的。归档区的

数据很少被访问，它能够被顺序地、不定期地、随机地访问。

除了不同的访问模式外，不同的区在数据量上也有很明显的差别。交互区的数据量相对较小。整合区的数据较多。如果一个企业中完全是近线数据，那么近线区通常会有相当大数量的数据。归档区的数据也可能显著增长，即使最初几年收集的归档数据相对较少，但随着时间的推移，大量数据完全有可能聚集到归档区。

难题出现了。在经典的数据仓库中，所有的数据都存放在磁盘存储器上，好像所有的数据都有平等的访问机会。但是，随着时间的推移和聚集的数据量的增大，数据的访问概率逐步下降，从而产生一种奇怪的现象：数据存放在磁盘存储器中越多，其被使用的次数越少。

成本昂贵，结果却得到较差的性能。事实上，成本是非常昂贵的。

性能不佳和高成本并不是第一代数据仓库达不到最佳的唯一原因，将数据划分成不同的生命周期区还有一些其他的合理原因，其中之一是不同的技术适用于不同的区。

2.5 元数据

以元数据为例。元数据是 DW2.0 环境下辅助的描述数据，用于告诉用户及分析员数据在哪里。图 2-4 说明了 DW2.0 构架下交互区与归档区元数据在处理上的显著差异。

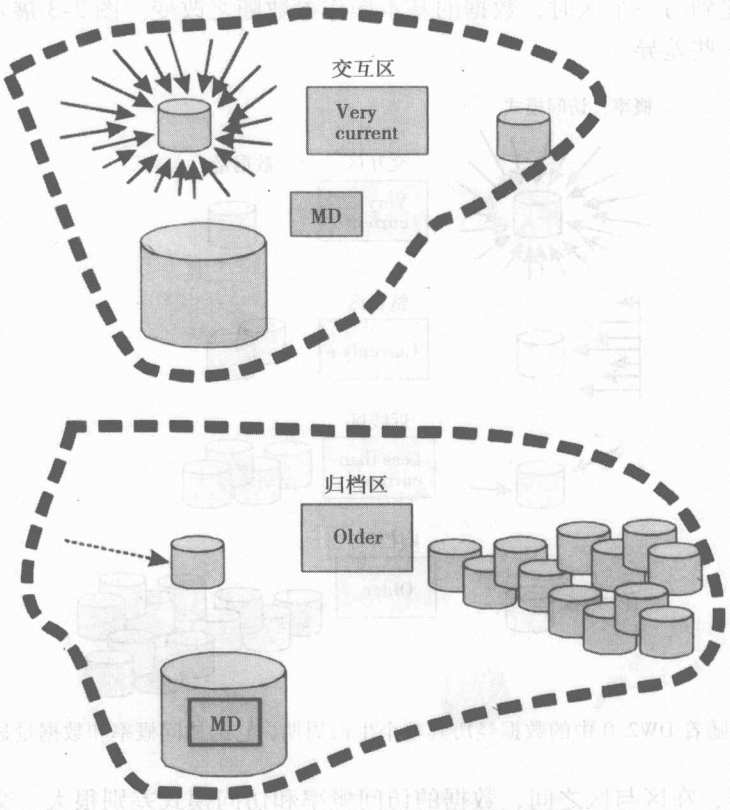


图 2-4 对于交互数据，元数据分开存储；对于归档数据，元数据直接与数据一起存储

图 2-4 表明，在实践惯例中是把元数据同实际数据本身分开存储。元数据存储在目录、索引、存储库及其他上百个地方。在所有情况下，元数据与它所描述的数据在物理上都是分开的。

相反，在归档区，元数据与它所描述的数据在物理上是一起存储的。归档环境中物理存储元数据的原因是归档数据可能 20 年或者 30 年没有使用过，谁知道它们将在未来的什么时候需要用到，或者用于何种目的？因此，元数据需要与实际数据一起存储，以便在审查归档数据时清楚地知道它是什么。为了更好地理解这一点，假设使用归档 30 年的数据。某人访问归档数据，惊奇地发现没有任何线索——没有罗塞塔石碑——来指示该数据的含义。在 30 年的时间里，元数据已经和实际数据分开，现在没人能找得到元数据。结果就是没人能够解释那一堆归档数据。

但是，如果元数据与实际数据本身物理地存储在一起，那么 30 年后当档案保管员打开数据时，实际数据的含义、格式和结构就会立刻清楚地呈现。

从终端用户的角度看，终端用户的满意度与元数据有关。有了元数据，终端用户可以判断数据和分析是否已经存在于企业中的某个地方。如果没有元数据，商业人士判断数据和分析是否已经存在就很困难。出于这个原因，元数据成为商业人士钟情 DW2.0 环境的因素之一。

2.6 数据访问

数据访问是 DW2.0 各个数据区之间的另一个根本区别。图 2-5 显示了数据访问的基本问题。

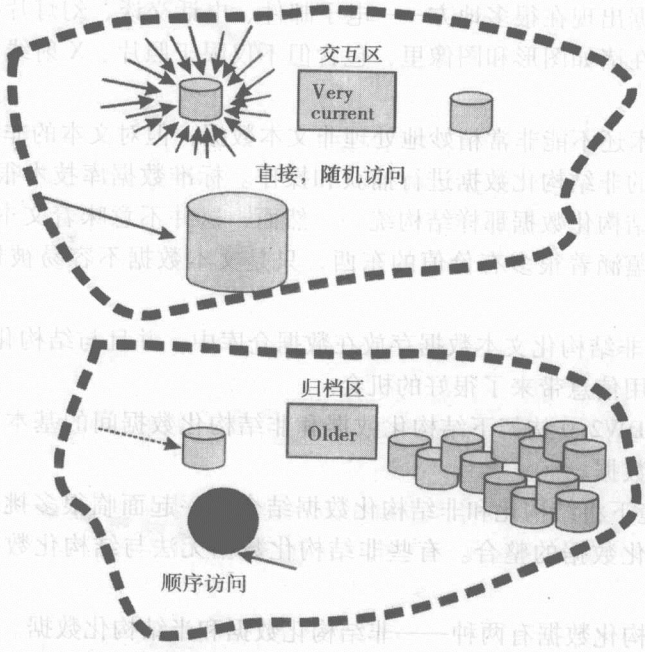


图 2-5 另一个主要区别是数据访问方式

该图强调了交互区与归档区在数据访问的方式与频率上的最根本区别。交互区的数据被随机地频繁访问。上一秒钟，一次交易发生，要求查看某一个单元的数据。下一秒钟，另一次交易发生，要求访问另一个完全不同的数据单元。并且这两次数据访问之后的处理都要求迅速访问数据，几乎是瞬间完成。

现在再来看归档数据的访问。就归档数据而言，很少对其访问。访问归档数据时，会顺序地访问其整段的记录。此外，归档数据访问的响应时间较为宽松。

我们也可以看到，在 DW2.0 构架下的各数据区之间，数据访问方式有很大不同，各区应用的技术也不同。因此，没有任何一种单一的技术——没有一刀切的技术——是现代数据仓库中发现数据的最佳技术。那种简单地认为只要把数据存储在磁盘上，一切都自己照顾自己的旧观念是不正确的。

尽管数据生命周期是 DW2.0 的一个重要方面，但它并不是其区别于第一代数据仓库的唯一不同之处。另一个主要区别是 DW2.0 环境中既包含结构化数据，也包含非结构化数据。

2.7 结构化数据/非结构化数据

一般存在两种基本类型的数据——结构化数据和非结构化数据。结构化数据表现为相同的格式和布局，通常由交易的保障来实现。结构化数据的典型例子包括银行交易、航空定位处理、保险交易、制造交易、零售交易等生成的数据。结构化数据便于存储在资料库记录中，记录中包含属性、键、索引、表格等。事实上，整个结构化数据世界得到了标准的数据库技术的较好支持。

另一类数据是非结构化数据。非结构化数据有两种基本形式——文本的和非文本的。文本的非结构化数据出现在很多地方——电子邮件、电话交谈、幻灯片演示等。非文本的非结构化数据出现在诸如图形和图像里，包含但不仅限于照片、X 射线、核磁共振、图表和插图等。

虽然目前的技术还不能非常精妙地处理非文本数据，但对文本的非结构化数据来说却不同。可以对文本的非结构化数据进行捕获和操作。标准数据库技术很难处理文本数据，因为文本数据不像结构化数据那样结构统一。然而，这并不意味着文本数据价值不够大。相反，文本数据中蕴涵着很多有价值的东西，只是文本数据不容易被标准数据库技术处理和操作。

DW2.0 要求将非结构化文本数据存放在数据仓库中，并且与结构化数据整合在一起。这为我们创新地利用信息带来了很好的机会。

图 2-6 显示了 DW2.0 构架下结构化数据和非结构化数据间的基本区分，并强调必须合并和整合这两种数据。

在 DW2.0 环境下将结构化和非结构化数据结合在一起面临很多挑战。其中之一是结构化数据和非结构化数据的整合。有些非结构化数据无法与结构化数据整合，而其他的可以。

事实上，非结构化数据有两种——非结构化数据和半结构化数据。非结构化数据是以自由的形式书写的文本。一本书、一本手册或者一个培训课程往往有大量的非结构化文本。半结构化数据是其中有重复性格式的文本数据。例如，一本食谱书是一个单一的文

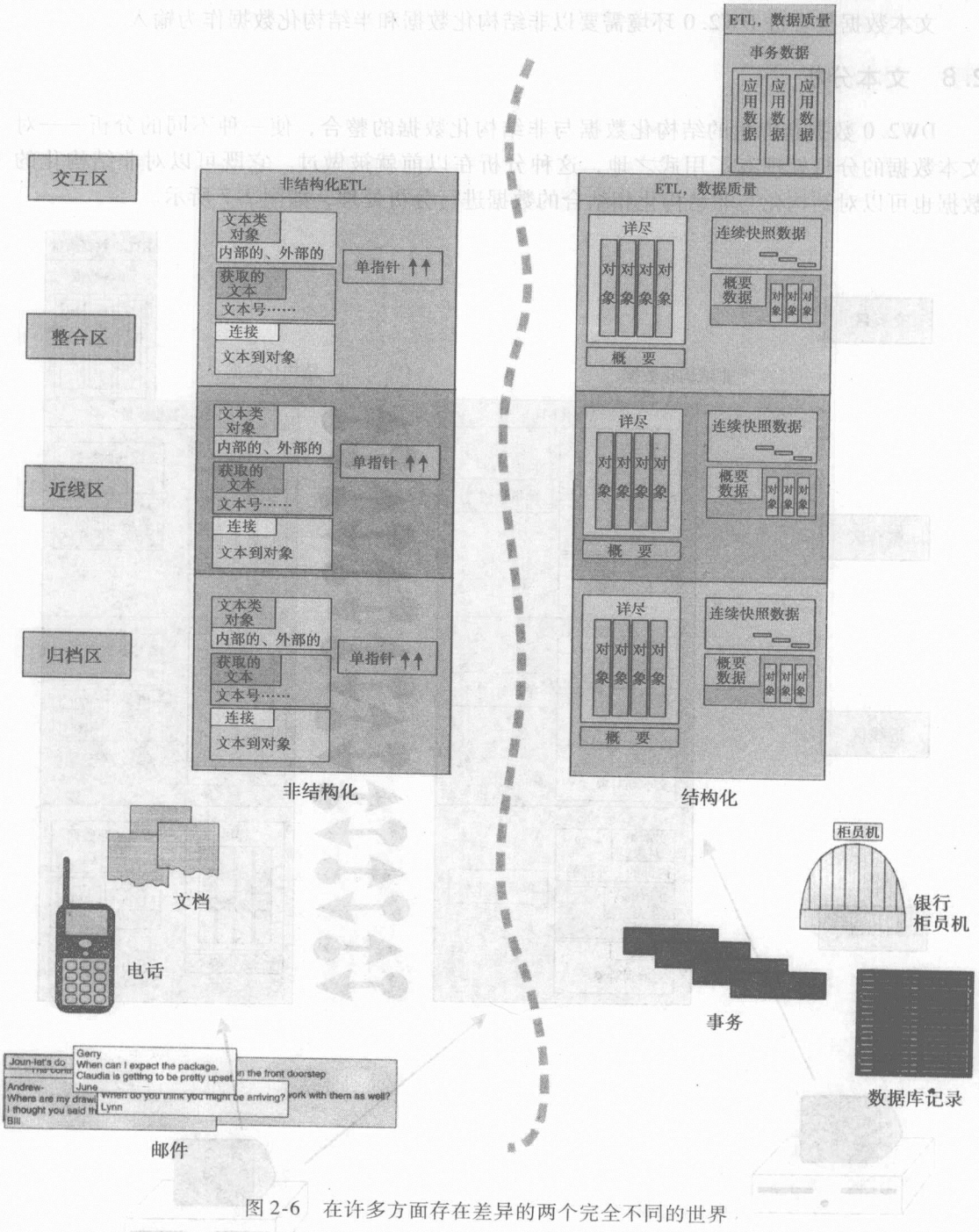


图 2-6 在许多方面存在差异的两个完全不同的世界

件，但在食谱书中有很多食谱，每一个食谱都有自己的材料和做法，每个食谱就是一个半结构化数据。

文本数据整合进 DW2.0 环境需要以非结构化数据和半结构化数据作为输入。

2.8 文本分析

DW2.0 数据仓库里的结构化数据与非结构化数据的整合，使一种不同的分析——对文本数据的分析处理有了用武之地，这种分析在以前就被做过。它既可以对非结构化的数据也可以对结构化与非结构化相结合的数据进行分析处理，如图 2-7 所示。

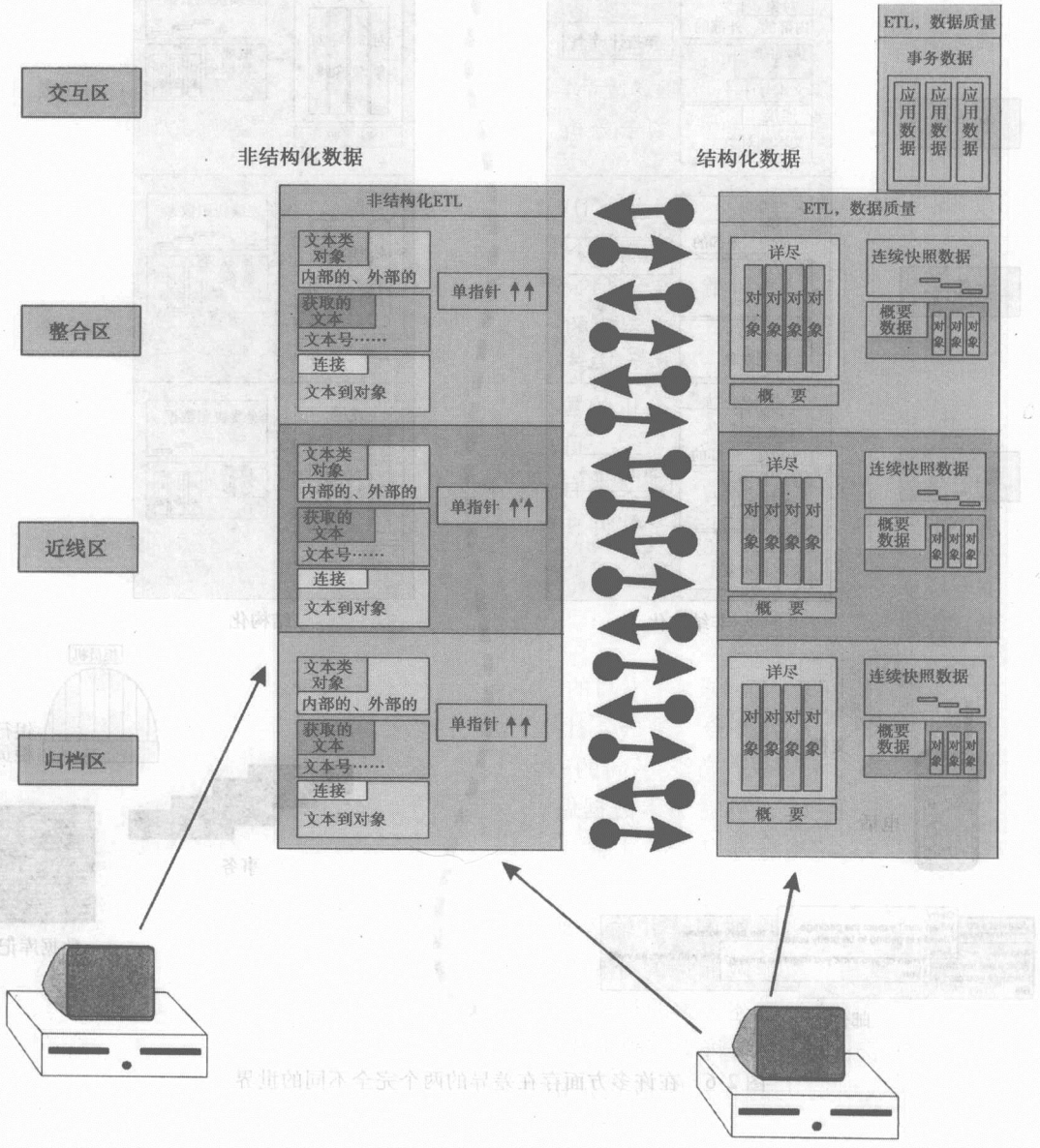


图 2-7 当数据仓库中包含非结构化的数据：a）非结构化数据单独进行访问分析；b）非结构化数据可以同结构化数据一起分析；c）一些非结构化数据可以近似地与结构化数据连接

2.9 “废话”

在 DW2.0 数据仓库中将非结构化数据与结构化数据合并面临着很多的挑战，其中之一是筛选非结构化数据。由于各种原因，非结构化数据中包含有一些可以称之为“废话”的信息。“废话”是对公司业务没有任何意义的数据。一个典型的例子就是电子邮件中的“废话”。

假设一封电子邮件这样写道：“亲爱的，今天晚上吃什么？”电子邮件世界里充满了这样的个人内容，个人邮件与公司的业务没有关系，对业务没有影响。若某人保存电子邮件，他通常并不需要保存这些“废话”，它只是一种妨碍。

电子邮件并不是唯一需要筛选废话的非结构化数据，所有非结构化数据都需要进行“废话”筛选。如果公司不屏蔽“废话”，那么加载到 DW2.0 环境的非结构化数据可能都是不相关的臃肿的数据，也不利于分析。因此，筛选是收集和管理非结构化数据的一个重要过程。

图 2-8 描绘了非结构化数据进入 DW2.0 环境前的筛选过程。

筛选“废话”仅仅是为 DW2.0 环境准备非结构化数据需要做的众多步骤中的第一步。另一项主要工作是对非结构化数据建立一个一般的（规范化的）文本基础。

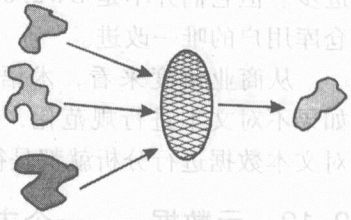


图 2-8 非结构化数据需要筛选

要想使非结构化数据对分析有用，必须将它转变成一种既能做一般分析又能做特殊分析的数据。想要理解一般和特殊基础对非结构化文本的需求，可以这样考虑，在几乎所有的公司任务中，非结构化文本有多种来源。无论是什么样的非结构化数据，不管是文件、电子邮件还是病历等，都是不同的人写的。

2.10 术语问题

由于文本是由许多不同的人书写的，因此必须考虑不同的人使用不同术语的问题。

由于不同的背景、年龄、种族、社会阶层、教育水平、国籍、母语以及许多其他因素，人们对同样的事情有许多不同的表达方式。如果对同样的事情的不同表达方式不“规范化”，那么就不可能对文本数据做出有意义的分析。因此，如果将文本数据用于文本分析，它必须首先经过规范化过程。

在进行文本分析之前需要对文本进行规范化还有另一个原因。这就是当作者写下文本时可能不知道该文本将被阅读并输入到数据仓库，因此作者没有考虑任何术语的解析。负责为数据库输入文本做准备的分析师需要阅读文本，并向吸收文本的系统描述这些文本如何解释。对机器描述文本的过程并不容易，这也是文本规范化所必需的一部分。

文本规范化过程要求将文字映射成两种格式——特定格式和一般格式。特定格式通常是人所说的或所写下的。一般格式是由人所说的或写下的东西经过规范化所达到的数据值。

举一个数据的一般格式和特定格式的例子。假设某人如下写道：“脚踝一直受到压力并且已经脱落。”这段话是一个具体的原始的文本。现在假设有人正在做一个关于骨头损

伤的研究。当其以骨头为关键字搜索时就不会搜索到脚踝字样，而以受伤为关键字搜索也不会出现脱落字样。但是，如果这段文本已经过预处理转化为特殊与一般格式，词语脚踝可以被认定为骨头，词语脱落可以被认定为受伤的一种表达。由特定格式建立一般格式的过程中就会出现两个表达——脚踝/骨头，脱落/受伤。在特定格式与一般格式间建立关联，并且这两种格式都已经存入 DW2.0 环境后，分析师在非结构化数据中查找骨伤的时候就会立即找到脚踝脱落。

因此，为 DW2.0 环境准备非格式化数据的第二个主要步骤是读取特定数据，并对特定数据添加一般数据，使得数据适合分析。做不到这点就是对时间和机会的浪费。

2.11 特定文本/一般文本

图 2-9 说明了在数据进入 DW2.0 环境之前，读取特定数据并为其建立一个一般文本数据的一般解释的必要性。

数据仓库中对数据生命周期的认识以及包含非结构化数据是在第一代数据仓库上的重大进步。但它们并不是 DW2.0 提供给下一代数据仓库用户的唯一改进。

从商业角度来看，术语规范化是必需的。如果不对文本进行规范化，那么商业人士试图对文本数据进行分析就都是徒劳。

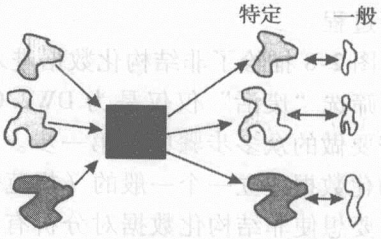


图 2-9 原始文本需要转换成特定/一般文本

2.12 元数据——一个主要组成部分

DW2.0 认为元数据是数据仓库基本结构的一个主要的并且极为重要的部分。

出于多种原因，元数据在第一代数据仓库中一直没有被认为或作为一个重要组成部分。随着新一代数据仓库的出现，元数据不再会被忽视。

元数据位于数据之上，用于描述实际数据中包含什么样的信息。元数据在任何环境下都很重要，而在 DW2.0 环境下尤为重要。

那么，为什么元数据对于 DW2.0 如此重要，而在第一代数据仓库中它却是可选的甚至被忽视？有一些非常好的、非常令人信服的原因可以说明元数据在 DW2.0 构架下的重要性：

- 规模和多样性：今天的数据仓库比以往的数据仓库更大，也更多样化。虽然以往的数据仓库可能已经能够非正式地掌握数据仓库中有些什么数据，但由于现在的数据仓库中数据的数据量和多样性，它不可能掌握其中所有内容。
- 更多样化的用户：现在的数据仓库的用户越来越多样化。过去只有少量的一些用户，他们形成了一个非常紧密的社区。而现在则有很多不同背景的用户。使这些用户了解数据仓库中有什么完全是数据仓库应该完成的工作。
- 广泛的元数据范围：元数据位于成功的 DSS（决策支持系统）处理的核心。要做到最优的决策支持系统处理，终端用户分析师必须知道很多关于可用于分析的数据的事情。终端用户分析师需要知道数据来自哪里、是什么意思、进行什么计算、包含哪些源数据、不包括哪些源数据、数据何时可用，等等。所有这些元数据信息对最终用户分析师都非常重要。

- 管理需要：随着数据仓库的成长，数据仓库环境的管理变得更加复杂。元数据越好，数据仓库环境的管理则越好、越简单。

这只是 DW2.0 包含元数据的几个原因。图 2-10 显示了 DW2.0 构架下的元数据层。

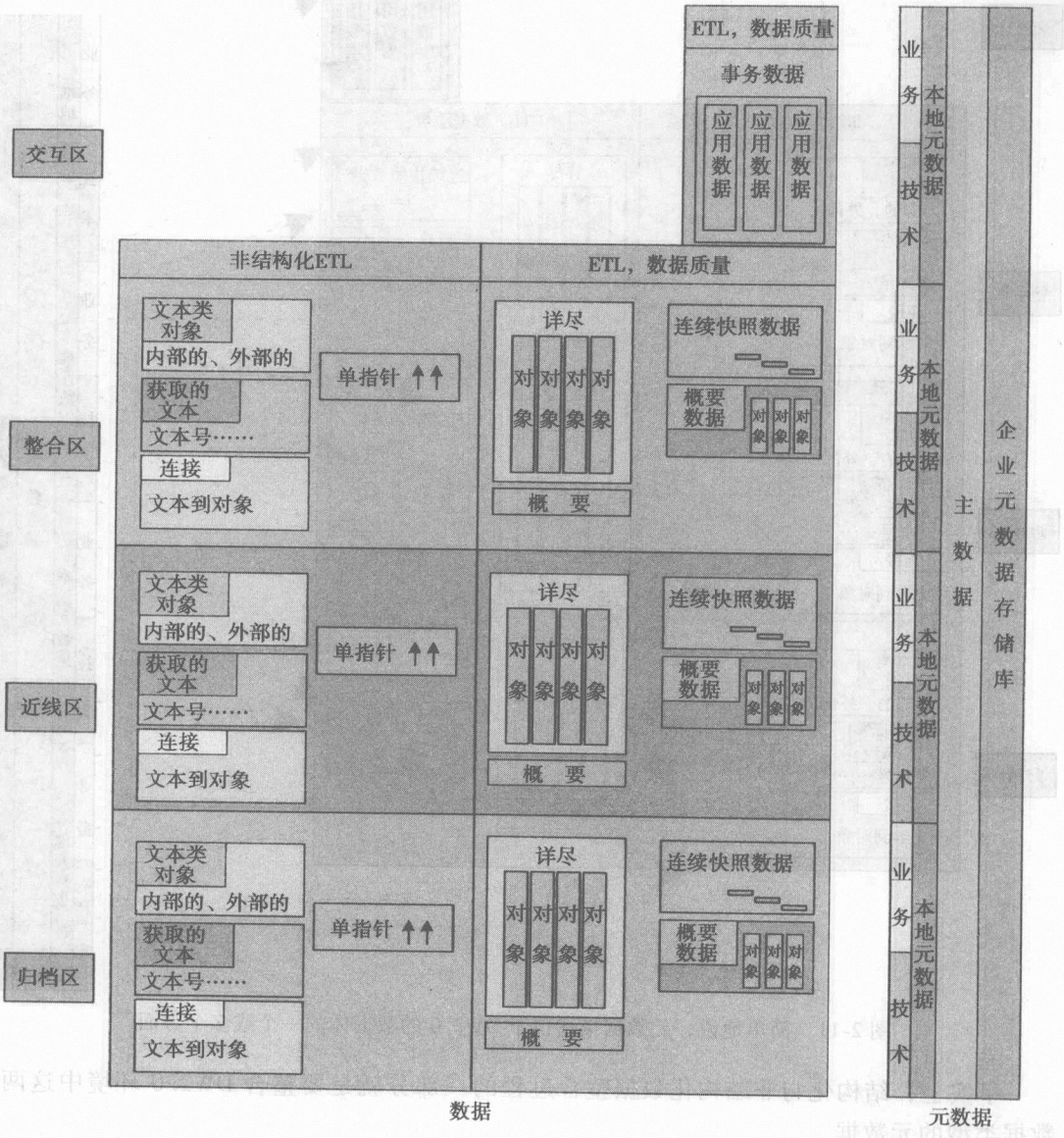


图 2-10　DW2.0 的另一重要组成部分是元数据

从某种意义上说，DW2.0 构架的元数据的组成很简单——至少在概念上。驻留在 DW2.0 环境下的元数据仅仅用来描述实际的 DW2.0 数据。

图 2-11 说明了 DW2.0 元数据观点。

值得指出的是，DW2.0 元数据需要描述结构化数据及非结构化数据。诚然，元数据的典型使用是用来描述结构化数据。但非结构化数据的引入，使得元数据在描述非结构化数据方面也非常有用。

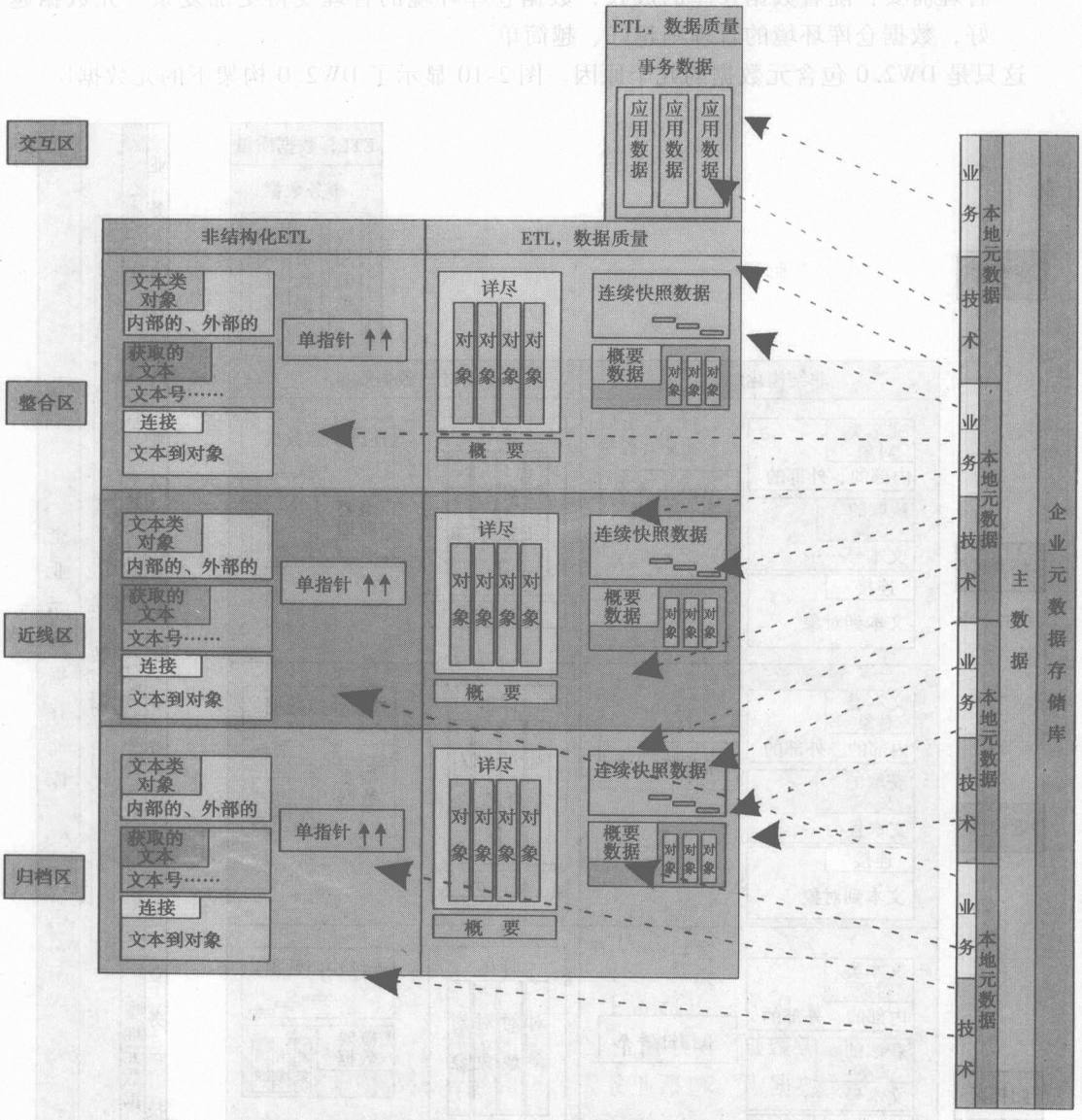


图 2-11 简单地说，元数据描述位于 DW2.0 的数据体的一个或多个方面

事实上，结构化与非结构化数据整合过程的一部分就是要整合 DW2.0 环境中这两种数据类型的元数据。

2.13 本地元数据

无论你前往任何会议听取发言，元数据的呼声都不绝于耳。事实上，今天我们周围有大量的元数据。ETL 工具中有元数据，数据库目录中有元数据，BI（商业智能）工具中有元数据……元数据无处不在！但是，这些元数据都是正在被使用的工具所专有的，可称其为“本地元数据”，而缺少的是企业范围的元数据。

DW2.0 需要这两层元数据——本地的和企业范围的。图 2-12 说明了因此产生的

DW2.0 的层次结构、关系及图表类型。

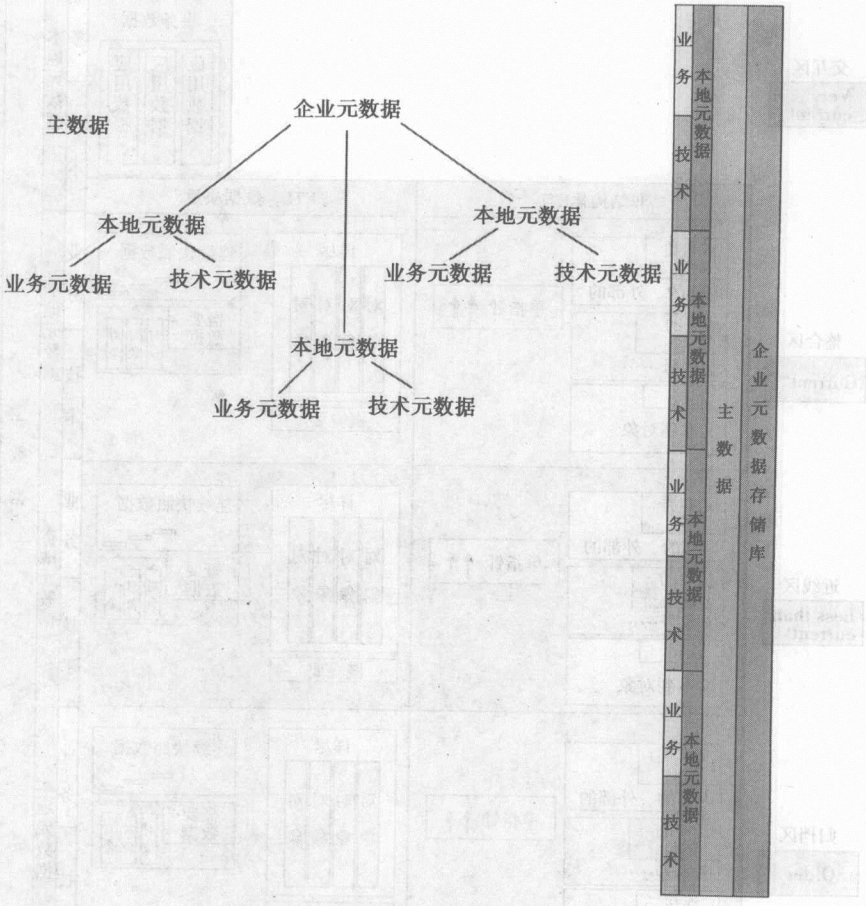


图 2-12 元数据环境的结构

图 2-12 表明 DW2.0 结构包括若干类型的元数据和一个整体的元数据结构。DW2.0 包括主元数据或“参考数据”。存在企业范围的元数据，也有本地元数据，就像给定工具中的那样。在本地元数据中，包括业务元数据和技术元数据。

业务元数据是用业务语言书写的适合于公司业务的元数据。技术元数据（大多数人所熟悉的那种）是公司的技术员所应用的元数据。

可以说，现在大多数的企业元数据与技术相关，而不是与业务相关。

DW2.0 在第一代数据仓库的基础上提出一些重要的改进。DW2.0 提出对数据进入数据仓库后的数据生命周期的认识、数据仓库中结构化与非结构化数据的概念以及元数据是数据仓库的一个重要的标准的组成部分的思想。

从第一代数据仓库环境到 DW2.0 还有另一个重要变化，即数据仓库的技术基础不应以不易改变的技术为背景的观点。

2.14 基础技术

图 2-13 显示了隐藏于每一个数据仓库下面的技术基础。

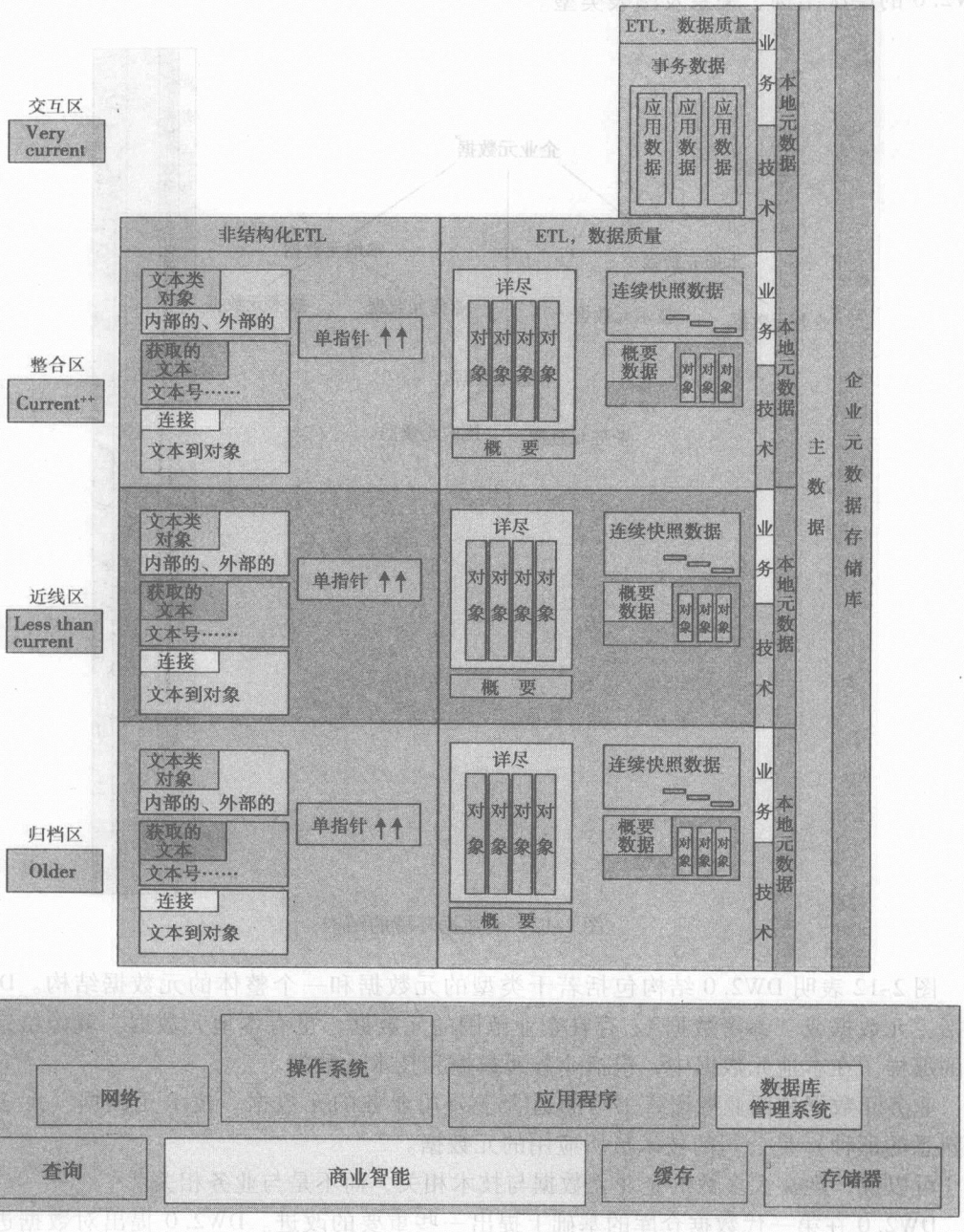


图 2-13 隐藏于 DW2.0 下面的是技术基础

技术基础是任何数据仓库的一个重要组成部分。简单地说，没有一个潜在的技术基础，数据仓库就不能存在。不仅对于 DW2.0 是这样，对于第一代数据仓库也是如此。

但是还有一个问题。基于数据仓库的技术使数据仓库根植于一种静止状态。一旦数据仓库深植于其技术，它就很难改变。这种技术往往会一成不变地构建数据仓库。只要数据仓库的需求没有改变，这就不是一个问题。当它需要改变或更新时，根除和/或改变它的技术基础这一极度困难的任务便成为一个重要的问题。

新一代 DW2.0 方法要求数据仓库不能一建到底。为了考察这种可能性，如图 2-14 所示，我们认为一个数据仓库的开发是通过围绕业务需求的逐步改造完成的。

不管怎样，数据仓库是为满足一系列的业务需求或要求而建立的。现在的问题是，随着时间的推移，这些要求因许多因素而改变，例如：

- 新的、修订的或解释的立法。
- 新的竞争。
- 经济环境。
- 新的技术。

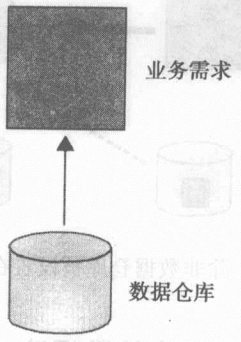


图 2-14 数据仓库是由创建它时的业务需求决定的

2.15 不断变化的业务需求

有人说只有两个东西永恒不变，即死亡与税收。但这里应该有第三个——业务变化。业务变化发生在每个人身上，尽管程度及方式不同。图 2-15 展示了业务需求随时间变化的情况。

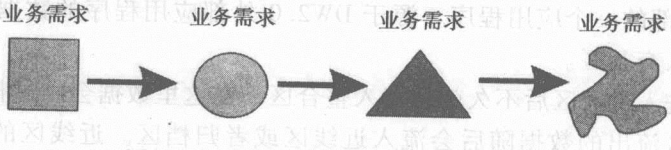


图 2-15 业务需求随着时间推移的变化

虽然业务变化是一种可预测的持续不断的现实，但是改变业务所依据的技术基础设施却是另一回事。出于多种原因，支撑大多数业务的技术像植根于混凝土一样，改变它相当困难。

因此，不断变化的业务与不变的技术基础设施是不相一致的。图 2-16 中建立了一个满足某种需求的数据仓库，用灰色立方体来表示。当业务的需求和条件变化时，它们应该变成灰色圆圈那样的形式。由数据所描述的技术基础设施原本是设计用来支持灰色立方体表示的商业需求的，但在业务发生变化后，这些技术基础设施却保持不变。

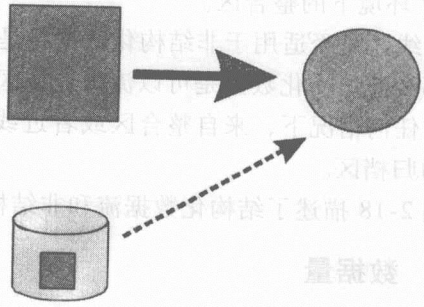


图 2-16 业务需求发生了变化，但数据仓库没有变

一个组织的数据仓库是其技术基础设施的很重要的并且往往是对业务非常关键的组成部分。现在的问题是数据仓库不容易改变，且设计时是为了满足早期的业务需求。技术基础设施一点点地变化，终有一天达到了原本的要求，但此时业务却又改变了。图 2-17 描绘了这种永无止境的不一致性。

因此，这让数据仓库构架师陷入了两难地步。而 DW2.0 解决了这一难题，它使用了以动态机制为基础的技术，可以很容易地随着时间的改变而改变。

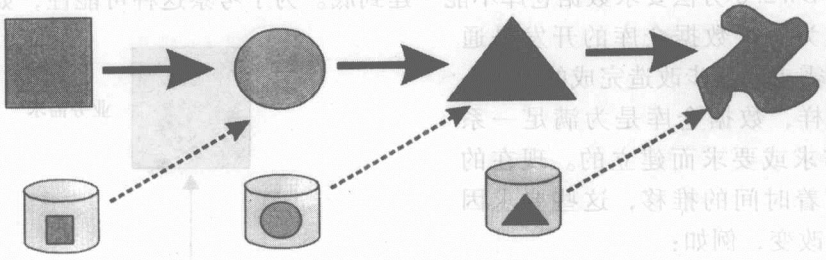


图 2-17 除非数据仓库被设置在一个动态技术基础上，否则它只是不断反映昨天的业务需求

2.16 DW2.0 中的数据流

用于描绘 DW2.0 的简图显示了其组成部分及它们彼此间的近似关系。没有显示出来的是在 DW2.0 环境下的整体的数据流。

虽然几乎所有的数据单元都有可能流向任何一个地方，但是对大部分的数据来说，一般都会得出一个其在 DW2.0 中可预测的数据流。

对于结构化处理，数据或者直接通过一个交互环境下的应用程序进入系统，或者来自于 DW2.0 环境外部的一个应用程序。源于 DW2.0 外部应用程序的数据会先通过 ETL 接口加工，然后流入交互区。

结构化数据进入交互区后不久就会流入整合区，在这里数据会经过整合并且转化为企业数据。从整合区流出的数据随后会流入近线区或者归档区。近线区的数据最终也会流入归档区。

非结构化数据采取近似的过程。非结构化数据是以文件或者某些其他格式的文本数据开始的，文本数据经历一个非结构化数据的 ETL 处理。然后，非结构化数据会进入 DW2.0 环境下的整合区。

近线区是否适用于非结构化数据仍是未知。但是，如果需要非结构化环境下的近线数据，那么非结构化数据是可以流入近线区的。

在任何情况下，来自整合区或者近线区（如果使用过）的数据流都能流入非结构化数据的归档区。

图 2-18 描述了结构化数据流和非结构化数据流经过 DW2.0 环境的情况。

2.17 数据量

另一种看待 DW2.0 环境中结构化和非结构化数据的有趣方式是从数据量的角度来看的，如图 2-19 所示。

DW2.0 环境的结构化部分的交互数据通常相对较少，而结构化整合数据量则有相当大的增长。当使用近线区时，它必须支持的结构化数据量会进一步增加。然后，当归档区比较成熟后，归档的结构化数据量还会有明显的增加。

相比之下，非结构化环境中整个生命周期区的数据量总是比结构化环境下的数据量更大，增长率也更高。据估计，在一些典型的企业中，非结构化数据是结构化数据的 4～5 倍。就算把无用的数据排除在外，非结构化的数据依然是结构化数据的 2～3 倍。这些数

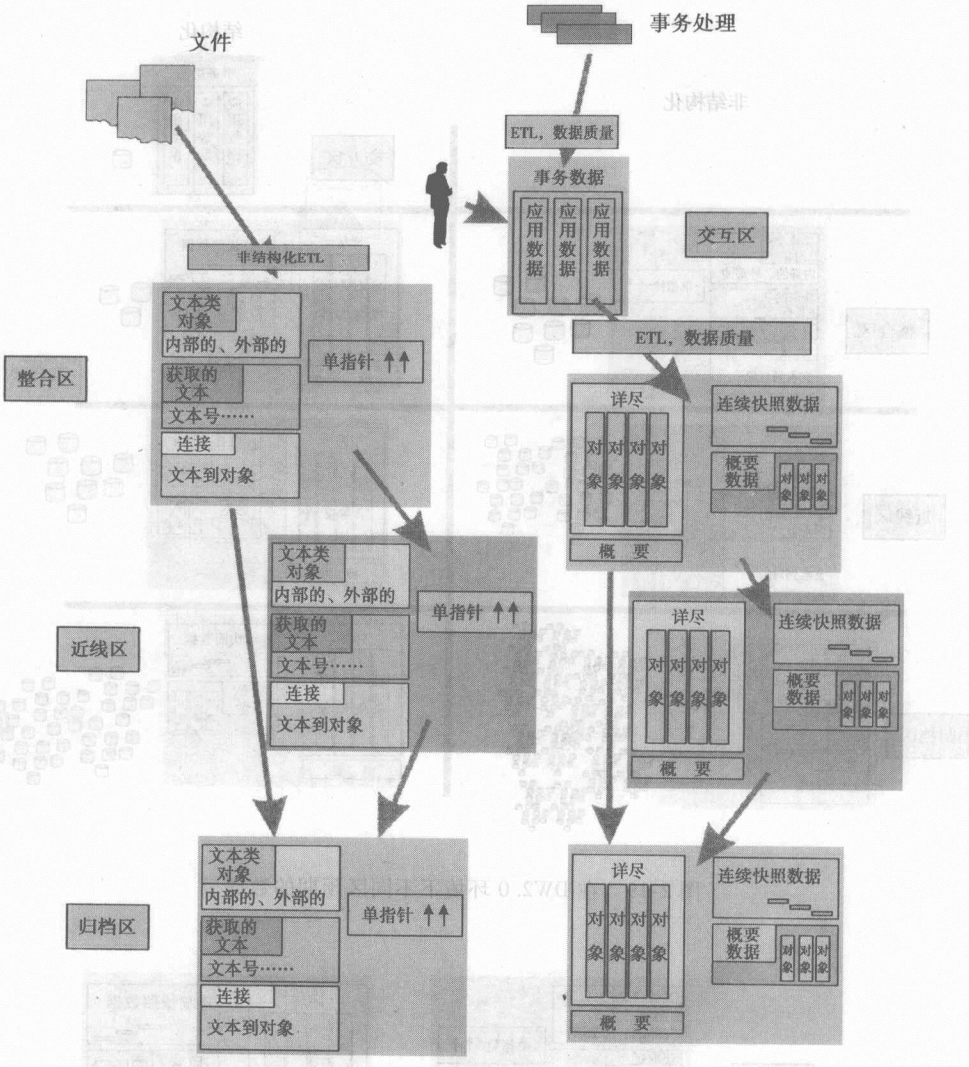


图 2-18 DW2.0 环境下的一般数据流

据量的比例关系也能从图 2-19 中看出。

2.18 实用应用程序

有些新的应用过去无法实现，而 DW2.0 提供了这种可能。DW2.0 对数据仓库的结构化和非结构化数据的支持引发了一些将这两种数据混合起来的有趣应用。

如图 2-20 所示，例如，当电子邮件的非结构化通信在数据仓库中遇见源于结构化数据的统计数据，有可能——第一次——创建一个真正的 360°的客户视角。

另一种有效的关于非结构化和结构化数据环境相结合的事例是医生的非结构化的笔记满足实验室的测试效果。这样，一份完整的、综合的电子病历就成为可能的、实用的并且易于实现的，如图 2-21 所示。

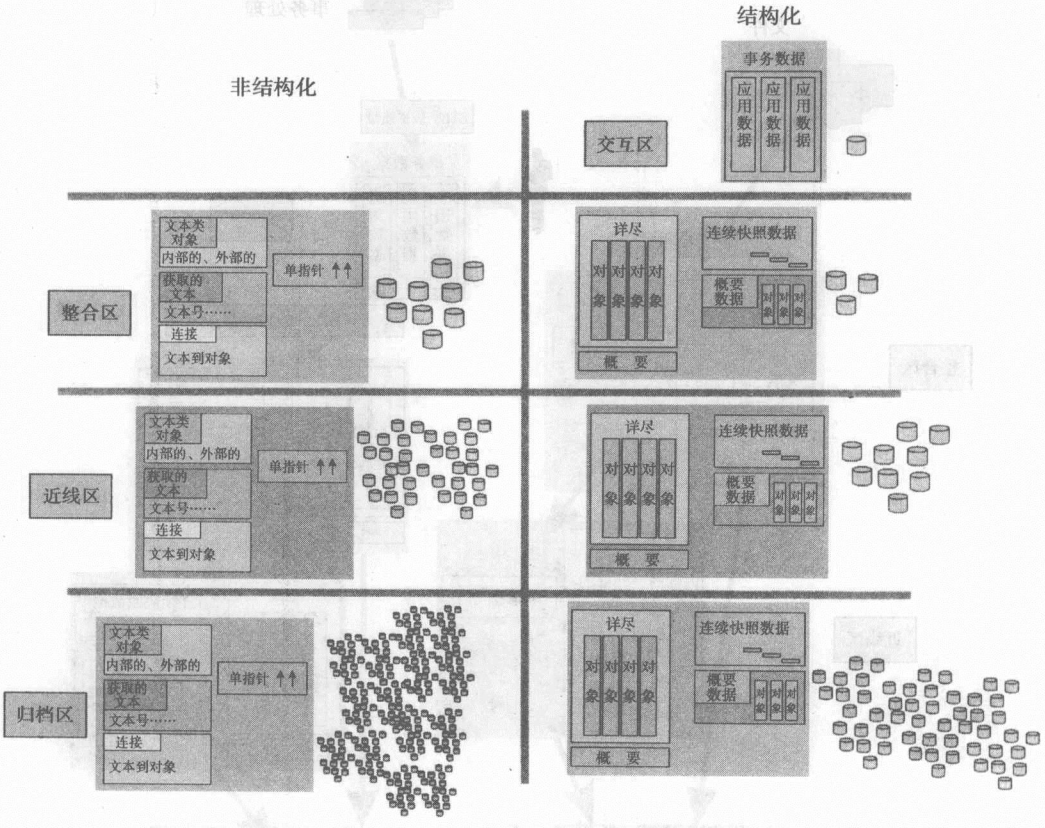


图 2-19 在 DW2.0 环境下不同区预期的数据量

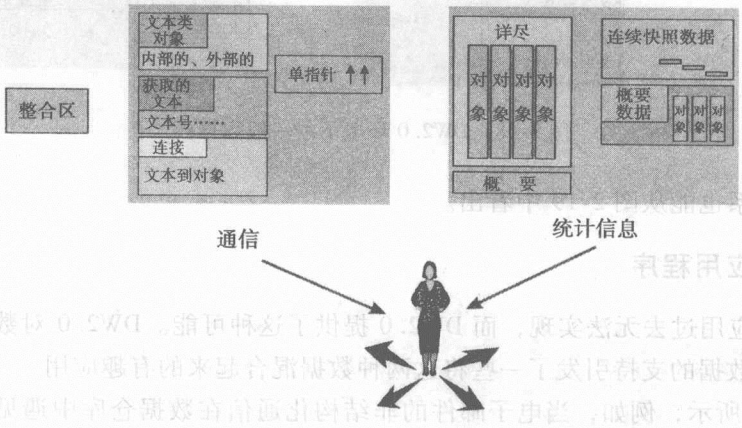


图 2-20 一个真正的360°客户视角

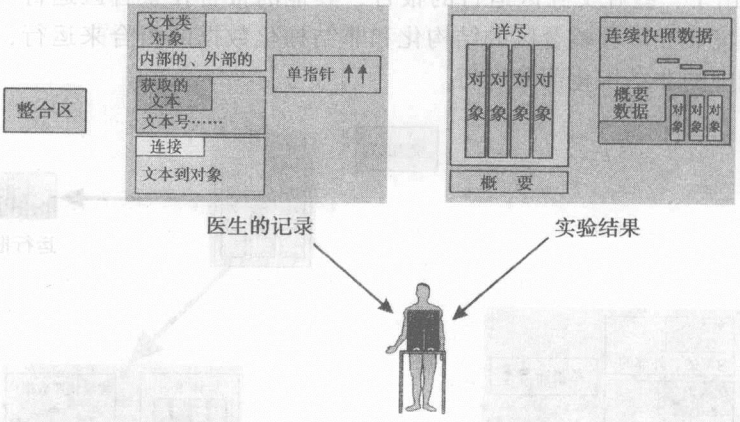


图 2-21 建立个人医疗保健记录

2.19 DW2.0 和参照完整性

参照完整性在数据仓库领域起重要作用已经有很长一段时间了。参照完整性要求居于一个数据库的数据也必须由一套符合逻辑的规则所控制。例如，如果医疗程序是分娩，那么病人的性别就必须是女性。又如，如果有购买发生，那么必定有一个产品或服务被购买。

DW2.0 方法延伸了参照完整性的概念。在 DW2.0 中，有外部参照完整性和内部参照完整性。图 2-22 给出了两种不同类型的参照完整性。

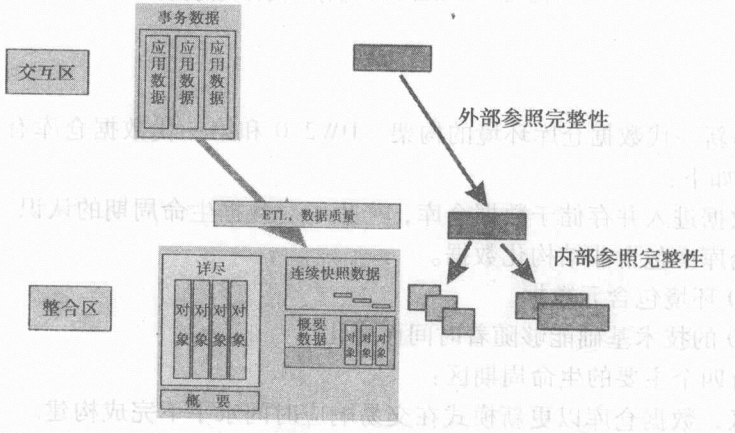


图 2-22 在 DW2.0 环境中有两种形式的参照完整性——外部的与内部的

如图 2-22 所示，外部参照完整性是指数据从一个区进入另一个区时完整性的保持，内部参照完整性是指数据在一个区内的完整性的保持。

2.20 DW2.0 的报告

几乎在 DW2.0 环境中的任何地方都可能出现报告。并非所有报告都在某一处运行，而是各种不同的报告在不同的地方运行。

图 2-23 给出了一些在交互区运行的报告，其他的报告在整合区运行。有些报告利用来自 DW2.0 环境交互区和整合区的结构化和非结构化数据的组合来运行，其他报告则通过 DW2.0 环境中的非结构化部分运行。

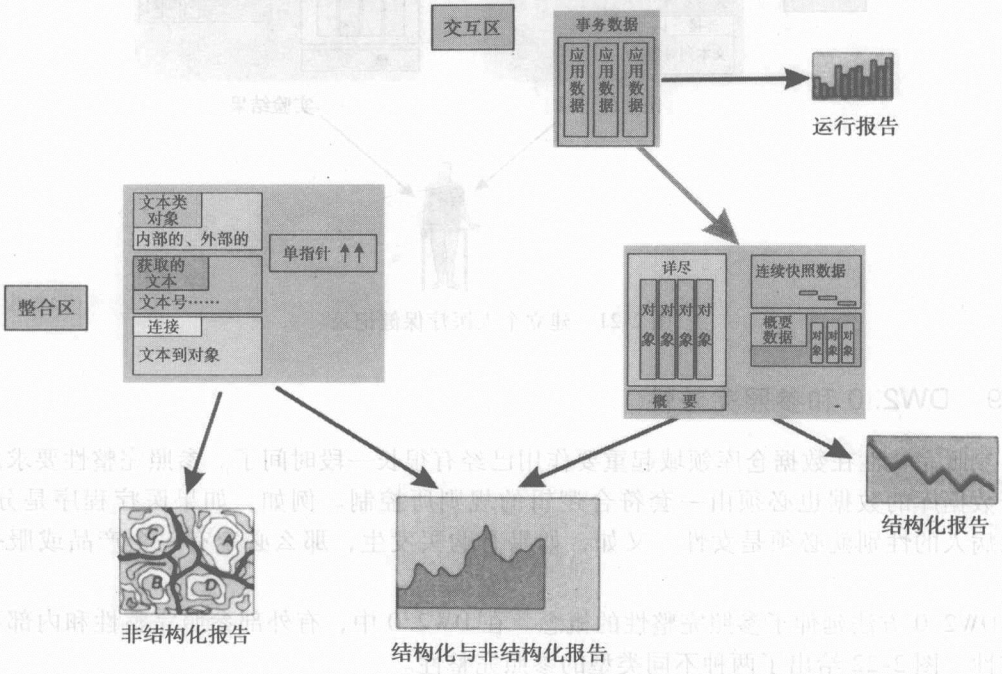


图 2-23 DW2.0 环境有不同种类的报告

2.21 总结

DW2.0 是新一代数据仓库环境的构架。DW2.0 和第一代数据仓库有很大的差别。四个最大的差别如下：

- 随着数据进入并存储于数据仓库，产生了对数据生命周期的认识。
- 数据仓库中包含非结构化数据。
- DW2.0 环境包含元数据。
- DW2.0 的技术基础能够随着时间而变化。

DW2.0 有四个主要的生命周期区：

- 交互区，数据仓库以更新模式在交易响应时间水平下完成构建。
- 整合区，数据在这里经过整合并完成分析处理。
- 近线区，作为整合区数据的一个缓存区域。
- 归档区，存放访问概率显著下降但仍有可能被访问的数据。

DW2.0 既包含结构化数据，也包含非结构化数据。非结构化文本进入数据仓库之前必须首先经过一个整合过程。整合过程对于为文本分析提供非结构化文本非常必要。如果非结构化文本未经整合，就无法有效地进行文本分析。

为非结构化数据进入 DW2.0 环境做准备的主要任务之一就是清除“废话”。另一项必做的工作就是术语规范化。文本必须同时拥有特殊的和一般的两种参考，以便成功地

进行文本分析。

元数据是 DW2.0 的一个重要组成部分，它有下几个级别：

- 企业级
- 本地级
- 业务级
- 技术级

要想获得成功，DW2.0 环境必须建立在可随时间变化的技术基础之上。

第3章　DW2.0组成部分——关于不同区

DW2.0由四个不同的区组成：交互区、整合区、近线区以及归档区。一般会根据数据仓库的大小和使用阶段来确定使用哪个区或者不使用哪个区。例如，在数据仓库的早期不可能存在任何归档数据，小型数据仓库也许根本没有任何近线存储。并且，不同企业中DW2.0数据仓库的具体实现也大不相同。

每一个不同的区有自己的一些考虑因素和特性。实际上，即使是在同一个区内，对结构化数据和非结构化数据的考虑因素相差也很大。

从企业的角度看，通常情况下，不同类型的使用者会从各自的区中访问和分析数据。在很大范围内，办公室人员会使用交换区来完成日常工作；整合区可以间接地支持不同的管理层——从公司的初级管理者到公司董事长；分析团队经常使用近线区；而归档区则使用得较少，或被那些保险统计员和工程人员使用。

此外，还存在着不同的用户群会访问和使用DW2.0环境的不同区。

3.1　交互区

交互区是数据进入DW2.0环境的入口。数据要么通过处于DW2.0外部的ETL应用进入DW2.0，要么是作为交互区内部应用事务的一部分来处理。图3-1显示数据进入交互区的各种渠道。

正如图3-1所示，交互区可能包含多种应用，这些应用可能包含也可能不包含整合数据。交互区中的应用可以更新并且能够具有高性能的事务处理，通常以亚秒来计算。

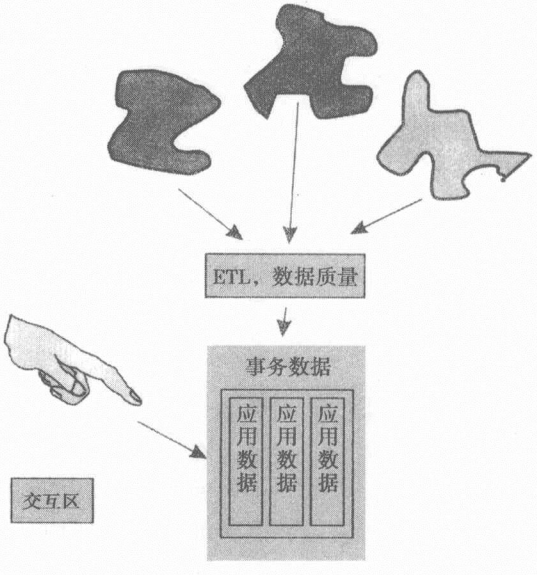


图3-1　交互区数据输入来源

通过交互区的事务流如图3-2所示。在图3-2中可以看到，在交互区存在着很多小型的事务流，这等同于某公路上仅允许保时捷和法拉利行驶，由于公路上没有行驶缓慢的车辆，交通工具的平均速度非常快，所以这个系统中运行的任何车辆的响应时间也都相当快。

交互区的另一个特征是由该区采用的技术所管理的数据量。交互区中仅有适量的数据，如图3-3所示。

交互区中运行的数据从几GB到几TB不等。相对于DW2.0环境的其他部分，交互区的数据量是比较小的。另外，交互区中的数据几乎总是存储在磁盘中。

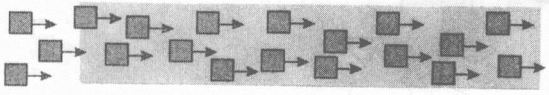


图 3-2 交互区内部的事务流

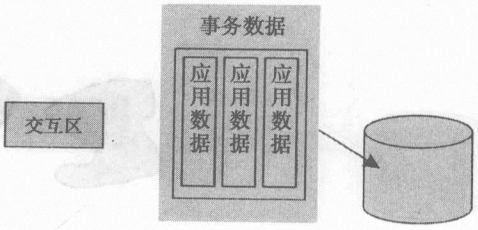


图 3-3 交互区中只管理少量数据

由于交互数据存储在磁盘上，并且交互工作任务通常是由较小且快速的事务处理组成，因此所有响应时间非常快。图 3-4 显示了标准情况下交互区的良好响应时间。

除了能获得好的性能外，交互区运行的事务处理还能更新数据。如图 3-5 所示，交互区的数据可以被添加、删除或者修改。

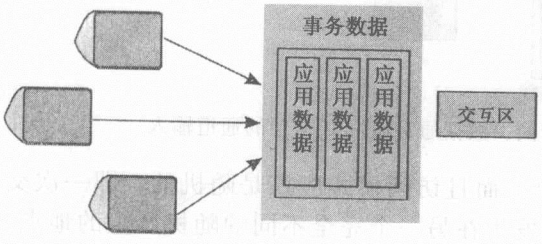


图 3-4 访问交互区的时间以秒计算

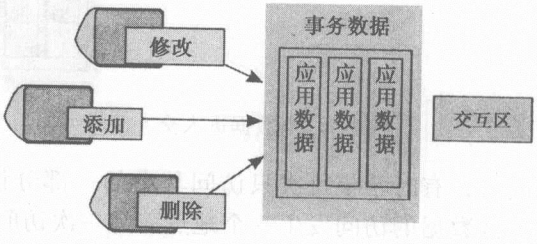


图 3-5 交互区中的处理可以更新数据

交互环境的一个特性是，由于数据可以更新，所以任何查询只在查询时刻有效。换句话说，假如在某时刻执行某查询，稍后执行同样的查询，那么这两次查询将得到不同的结果。如图 3-6 所示，在 10:31AM 执行一个查询获知账户余额为 $3000，而在 10:53AM 执行同样的查询时账户余额为 $4500。在两次查询之间有人已向账户增加储蓄。

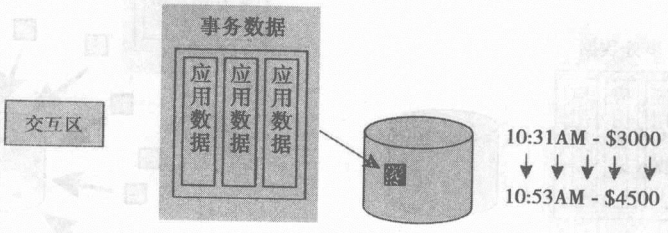


图 3-6 由于交互环境中的数据可以更新，所以任何访问只在访问的时刻正确

如果数据是通过外部应用程序进入交互区，那么数据需流经 ETL 层。换句话说，如果一个应用程序位于交互区之外，且要使这个应用程序的数据进入交互区，那么数据必须经过 ETL 工具的处理。图 3-7 显示了这一过程。

未单独使用 ETL 工具进行整合处理的数据是可以进入交互区的。这种情况下，数据在进入整合区的时候被整合处理。

交互区中的数据可能参照也可能不参照对其设置的约束。使不使用参照完整性完全取决于运行的应用程序。图 3-8 说明交互区可能包含也可能不包含可选择的参照完整性。

交互区中数据的访问特征是访问速度非常快——以亚秒为单位。当交互区的数据被访

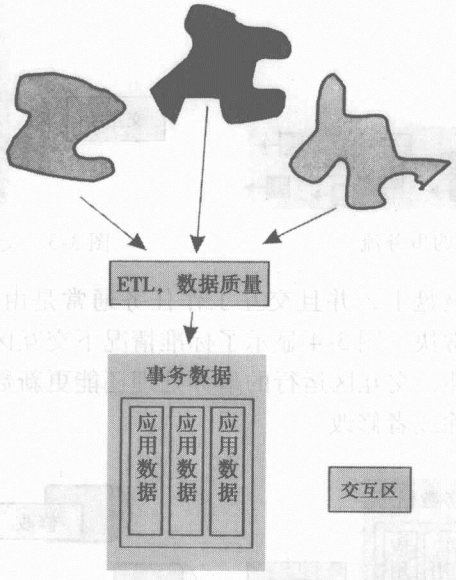


图 3-7 当数据进入交互区时可能需要转换，数据通常通过 ETL 层的通道输入

问时，有时希望每次只访问其中的一部分记录。而且访问模式应该是随机的，即一次交互区数据的访问发生一个地点而另一次访问则发生在另一个完全不同的随机产生的地点。交互区内部的这种数据访问模式决定了磁盘存储是理想的。

图 3-9 描述交互区中常见的迅速、随机的数据访问模式。

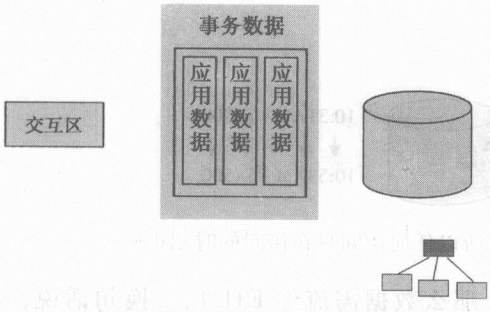


图 3-8 参照完整性可被强加在交互区的中间层

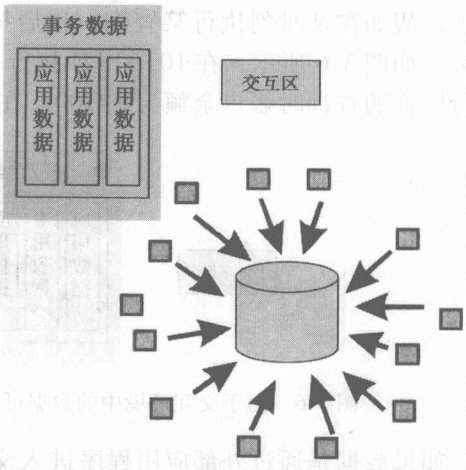


图 3-9 访问模式：a）快；b）随机；c）少量数据

交互区内部仅有少量的历史数据。在交互区中，典型的历史数据都是一天甚至仅仅是几小时以前的数据。一般情况下是找不到几个月以前的数据的，数据在变旧之前就已经进入整合区了。

图 3-10 展示了交互区没有维持太多的历史数据的概念。

交互区中的数据的粒度是非常不均匀的。一些应用程序使用的数据的粒度比较小，而另外一些应用程序则会整合数据，使其粒度非常大。如图 3-11 所示，针对应用数据的粒

度问题，不存在一个一致性的设计。

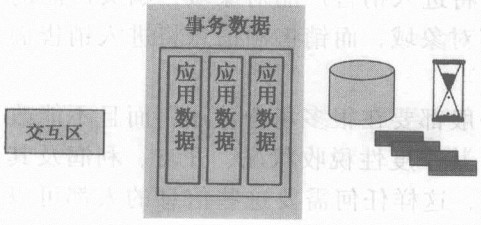


图 3-10 交互环境中仅有少量的历史数据

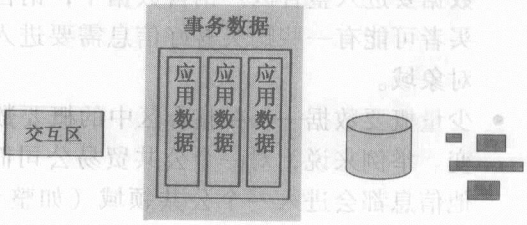


图 3-11 交互区内部的数据粒度依应用程序的不同而不同

数据从交互区进入整合区。如果数据来自交互区外部，则直接进入整合区。如果数据是由交互区内应用程序执行时产生的，那么数据将作为程序运行的副产品被收集并传给整合区。图 3-12 给出了所有进入或者通过交互区的数据最终输入整合区的过程。

3.2 整合区

整合区是应用程序数据和交易数据最后汇总为企业数据的场所。举一个应用程序数据和企业数据之间区别的例子，考虑三个收集税收信息的应用程序 A，B，C。程序 A 用欧元统计税收信息，程序 B 用美元统计信息，程序 C 用比索统计信息。当这些应用程序进入整合区时，所有的数据都将经过转换，使用一种通用的货币符号。

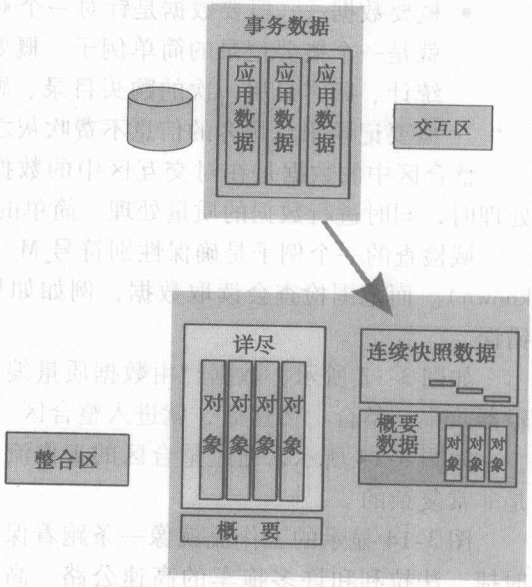


图 3-12 数据离开交互区的流向

把操作应用程序数据和交互交易数据转换成企业数据还需要做很多转换。例如，假如应用程序 A 进行统计时按每个月为 31 天、在第 31 天进行结算统计，应用程序 B 按自然月统计，而应用程序 C 则以企业日历为准。因此，来自应用程序的结算报告与企业结算报告不相符，它们可能有不同的实际结算日。只有在数据进入企业整合区时，将这三个结算日转化通用的结算日，这种不一致才会得以解决。

另一种调和是对数据关键字的调和。假如应用程序 A 有一种关键字结构，应用程序 B 有另外一种关键字结构，而应用程序 C 再有一种关键字结构。当来自这些应用程序的数据进入到整合区时，关键字会被转换成一种企业通用的关键字结构。

还有一个例子，应用程序 A 的日期格式是 YYMMDD，应用程序 B 的日期格式是 MMDDYY，应用程序 C 的日期格式是 YYYYMMDD。当日期被输入整合区时，这三种日期格式都被转换成一种通用的格式。

除此之外，数据由交互区进入整合区时还存在一些其他类型的转换的例子。

整合区包含多种不同的结构，以下几种类型的数据结构都可以在整合区中找到：

- 面向对象的详细数据——这种数据类型下，数据被组织成为较大的对象域并且保

存了详细细节。举个详细数据被组织成对象域的例子，假定一个销售交易的细节数据要进入整合区。销售数据中，销售条目将进入销售产品对象域，购买产品的买者可能有一些购买喜好信息需要进入顾客对象域，而销售额信息将进入销售额对象域。

- 少量概要数据——在整合区中的概要数据一般都要在很多场合使用，而且不能改变。举例来说，对一个公共贸易公司而言，其季度性税收状况、开支、利润及其他信息都会进入一个公共领域（如整合区），这样任何需要这些信息的人都可以访问到它。
- 持续时间跨度的数据——对于一些变化较慢的数据，将它们以连续时间跨度的结构来存放是很有用的。例如，连续时间跨度结构可用于记录顾客的姓名和地址。除非顾客的婚姻状况改变，否则他的地址和姓名不会经常改变。因此，在整合区中以连续的记录保存关于顾客的信息是可能而且是合理的。
- 概要数据——概要数据是针对一个对象从各种渠道收集到的数据。一个顾客记录就是一个概要记录的简单例子。概要记录用来记录顾客的信息，例如，顾客人数统计、顾客最后一次的购买目录、顾客活动的时间、顾客消费的地点，等等。从概要记录跟踪顾客的信息不费吹灰之力。

整合区中的数据是在对交互区中的数据通过 ETL 层处理后收集得到的。在进行 ETL 处理时，同时进行数据的质量处理。简单的数据质量处理就是域检查和范围检查。

域检查的一个例子是确保性别符号 M（男，male）、F（女，female）、U（未知，unknown）。而范围检查会读取数据，例如如果年龄大于 150，范围检查可能会将其标记为错误。

如图 3-13 所示，数据经由数据质量编辑器收集、整合、传递后，就进入整合区。

如图 3-14 所示，通过整合区的工作流是非常复杂的。

图 3-14 显示的工作流就像一条跑着保时捷、法拉利和许多拖车的高速公路。高速公路上车辆的速度取决于它前面的车辆。一辆时速 185mph 的保时捷如果在一辆时速 25mph 的拖车后面行驶，那它也只能跑 25mph。我们很容易看到，这种工作流与交互区的工作流相比，完全是两个级别。

整合区的复杂工作流也有其合适的理由。有些人需要访问大量数据，而一些人只访问少量的数据，就这么简单。但他们都想从整合区得到数据，所以导致非常复杂的工作流。

整合区通常包括大量的数据，如图 3-15所示。

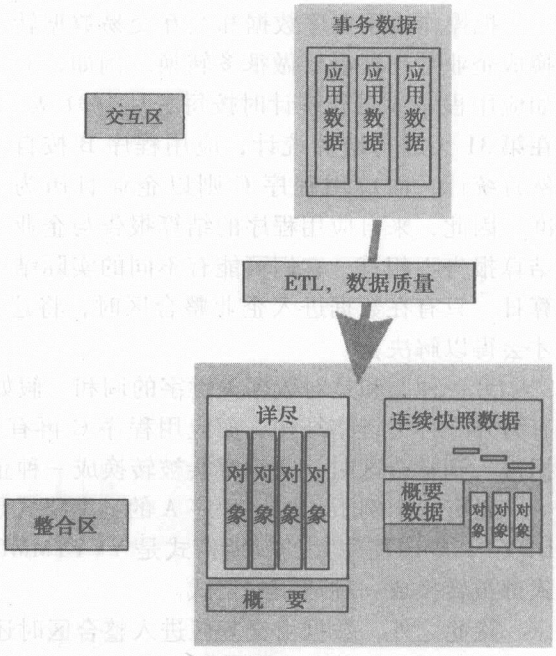


图 3-13 数据怎样进入整合区

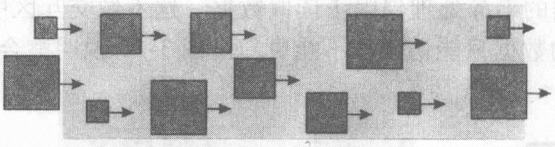


图 3-14 工作流在整合区内的运行演示

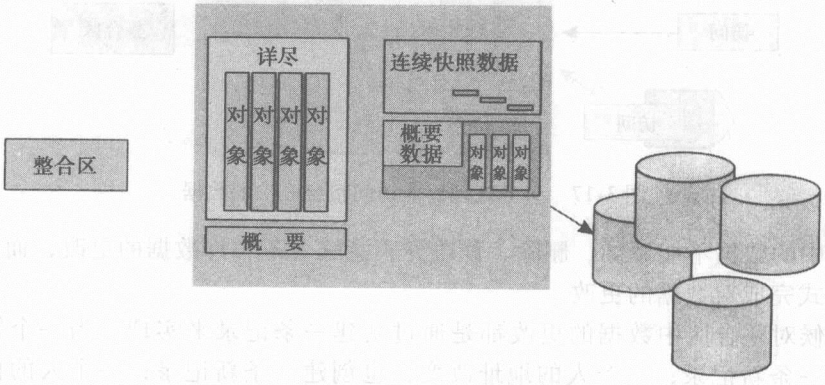


图 3-15 整合区管理着大量的数据

对于整合区为什么会包含大量的数据，有一些非常好的原因，包括：

- 数据是粒状的：很多原子单元的数据被收集和管理。
- 历史数据：经常有 3 ~ 5 年的有价值的数据。
- 数据来源于多种渠道。

将这三个原因综合起来，就会得到整合区中包含很多数据的结果。

整合区中复杂工作流的不同带来所希望的响应时间的不同，整合区中的响应时间从 10 秒到更长时间不等。不一致的原因就是混合的工作流。当整合区中进行大规模数据获取工作时，对少量数据的查询可能被暂停或者延迟。另一方面，当没有其他人使用时，访问整合区的用户能获得较好的响应时间。

控制响应时间的一个方法是不让拖车在高峰时间上路——在 8:00AM 至 3:30PM 时间段只让保时捷和法拉利在公路上行驶。如果这种方法奏效，那么就限制了拖车师傅上路行驶的合法需求。换句话说，通过将大的查询任务限制在空余时间段，那些需要在高峰期使用整合区数据的较小任务的响应时间就能够得到提高。相应地，大型查询用户将获得较差的响应。

至此可以看出，整合区中复杂工作流各种各样。图 3-16 描述了整合区响应时间的期望值。

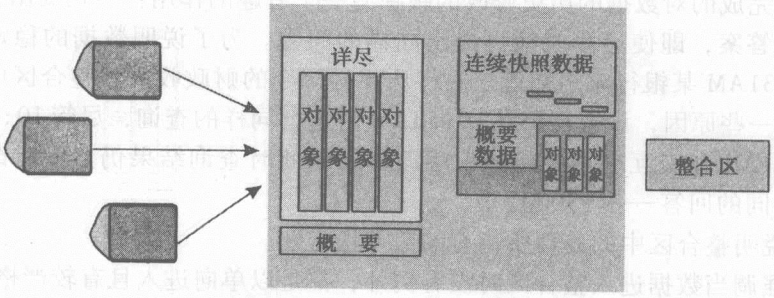


图 3-16 访问整合环境的响应时间从 10 秒到 2 小时或者更多

访问整合区中数据的事务处理仅限于读取数据。这不像交互区中，数据可以添加、删除、修改，整合区中的数据只能访问，不能更新。图 3-17 说明整合区中的数据访问。

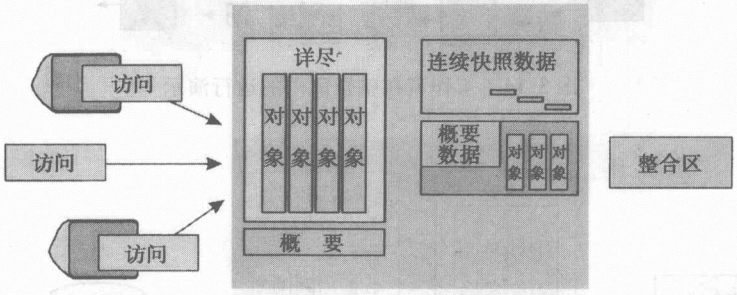


图 3-17 事务处理仅能访问整合区中的数据

整合区中的数据不能添加、删除、修改并不意味着不容许数据的更改，而是可以以一种不同的方式完成对数据的更改。

任何时候对整合区中数据的更改都是通过新建一条记录来实现。当一个银行账户改变，便创建一条新记录；一个人的地址改变，也创建一条新记录；一个人的保险内容改变，同样也创建一条新记录，等等。每次改变都通过创建一条新记录来完成，这样数据变化的历史跟踪记录也被保存下来。另外，数值被正确地放置在整合区中后就永远不能修改。记录可能被发送到近线存储和归档存储中，但一经正确创建后，就不能更改。这意味着改变的处理方式与交互区中的处理方式有很大不同，在交互区中，对一条记录的更改始终在进行。

图 3-18 说明整合区中更改是怎样被记录的。

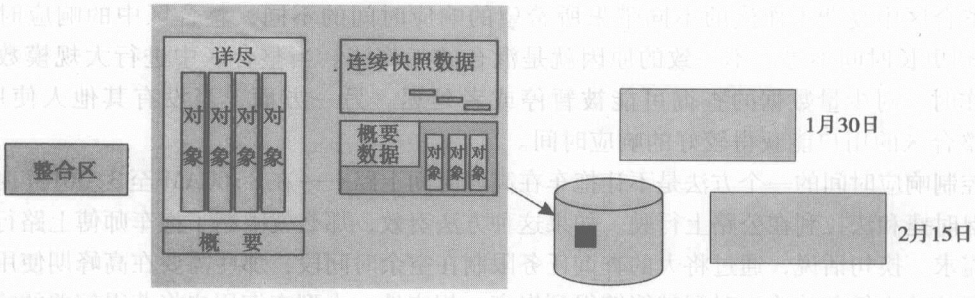


图 3-18 当整合区中的数值改变时，将创建并存储修改记录。因此存在修改的历史记录

整合区中完成的对数据的历史修改的跟踪有一个有趣的作用：一旦提出问题时，将总会得到同样的答案，即使过段时间再提出同样的问题。为了说明数据的稳定性，假设在今天早上 10:31AM 某银行家想知道该银行从年初至今的财政收入，整合区中的返回值为 $3000。出于一些原因，该银行家在 10:53AM 时进行同样的查询，尽管 10:31AM 银行的收入在银行的操作和交互应用中已经发生改变，但此时查询结果仍然没有改变，银行家仍然会得到相同的回答——$3000。

图 3-19 说明整合区中的处理所特有的数据稳定性。

图 3-20 强调当数据进入整合区时没有例外，都是以单向进入且有较严格的控制路线。

整合区中有两种相关的参照完整性。第一种是区间参照完整性，区间参照完整性指的

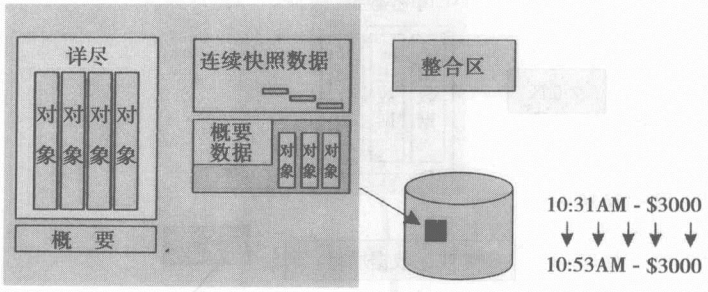


图 3-19 整合环境中的数据不能更新，任何时候访问的任何值都是正确的

是数据通过各区时的完整性。也就是说，当数据从交互区进入整合区时，数据必须有可辨别的源和目的以确保数据不会丢失。交互区中如果没有相应的数据输入则整合区中也没有数据输入——反过来也一样，整合区中没有相应的数据输入，则交互区中也没有数据输入。

然而，并不会仅因每个区有相对应的数据入口，就意味着所有的输入值就都应该是一致的。一个输入的值可能用欧元表示，而另一个输入的值可能用美元表示，两个数据元素没有相一致的值也不意味着它们就不是整合区相对应的输入。

整合区中的另一种参照完整性是相同区内的参照完整性，这种完整性意味着同一区内部数据元素之间可能存在某种关系。如图 3-21 所示，在整合区中，各种参照完整性都是可能存在的。

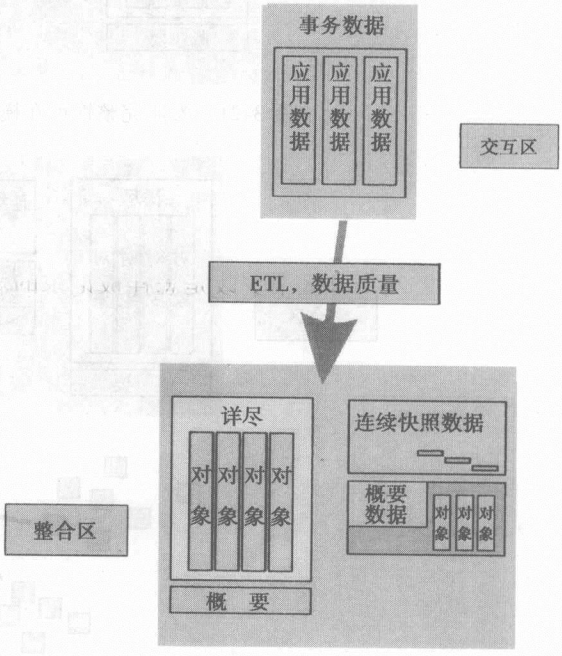


图 3-20 数据进入整合模块时通常先经过 ETL 处理后在综合

与交互区相比，整合区中数据的访问模式有所不同，对数据的调用较少，但每次调用常需要更多的数据。图 3-22 说明了对于整合区特定的访问模式。这种数据访问模式常常伴随着由从小到大的各种数据提取请求组成的复杂工作流。

整合区和交互区的另一个区别与不同环境中历史数据的容量有关。整合区中有大量的历史数据，在其中找到 3 ~ 5 年的有价值的历史数据是很正常的事情。相反，在交互区中，找到多于 30 天的有价值的数据都非常困难。

图 3-23 展示了整合区中大量的历史数据。

数据粒度也是整合区和交互区之间的另一个主要区别。交互区中有着不同的粒度级，一些数据是粒状的，而一些不是。另一方面，整合区包含公司中最小粒度级的数据，整合区中的数据非常小而且是原子的。整合区支持各种形式的 DSS（Decision Support Systems）

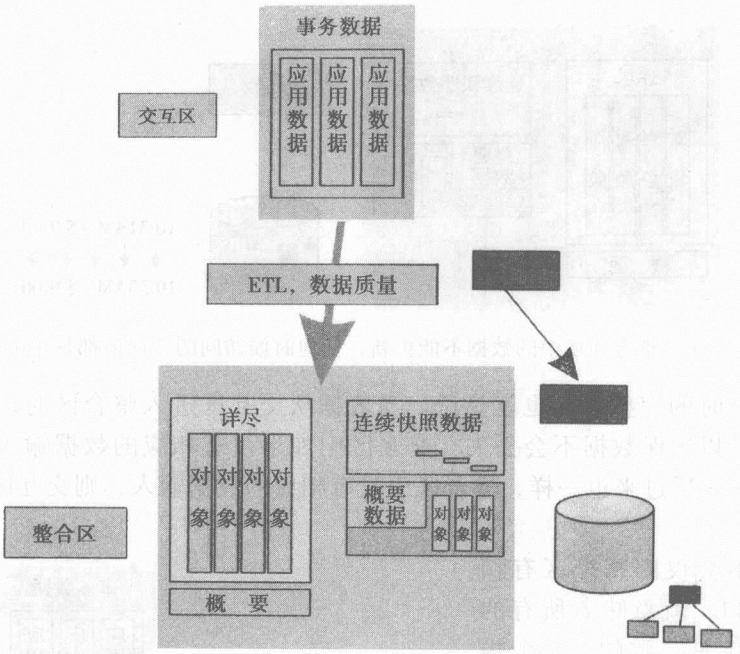


图 3-21 参照完整性可在模块内或模块间添加

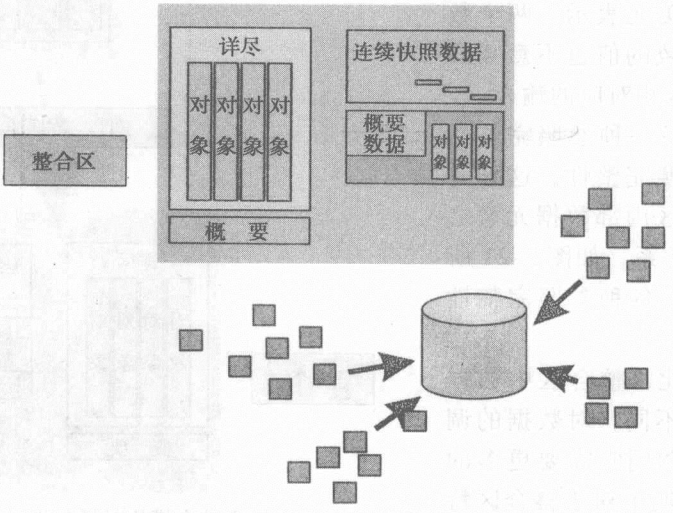


图 3-22 访问模式：a）严格速度；b）持续随机；c）大量数据

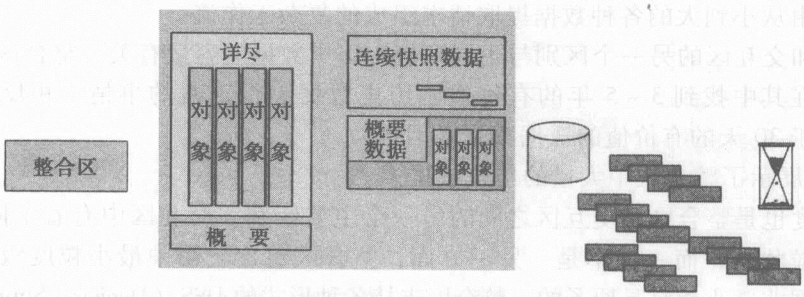


图 3-23 整合环境中有大量的历史数据

处理，每个DSS处理对数据都有自己的需求。因此，整合层次上粒度级越小，所支持的DSS处理的形式就越多。换句话说，当整合区中数据粒度级变大时 可支持的DSS处理的形式则越少。

图3-24描述了整合区中对低粒度级的需求。

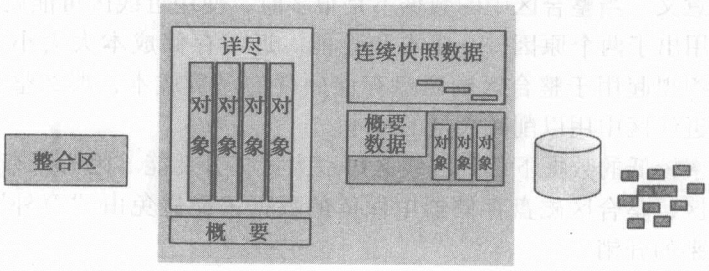


图3-24 整合区中的数据粒度和从信息系统环境中得到的一样小

数据一旦离开整合区，就可能进入近线区或者归档区（图3-25）。当数据很多且有缓存需求时，数据便进入近线区。当数据的访问概率显著降低时，数据进入归档区。通常，随着数据变陈旧它将进入归档区，但并不总是这样。

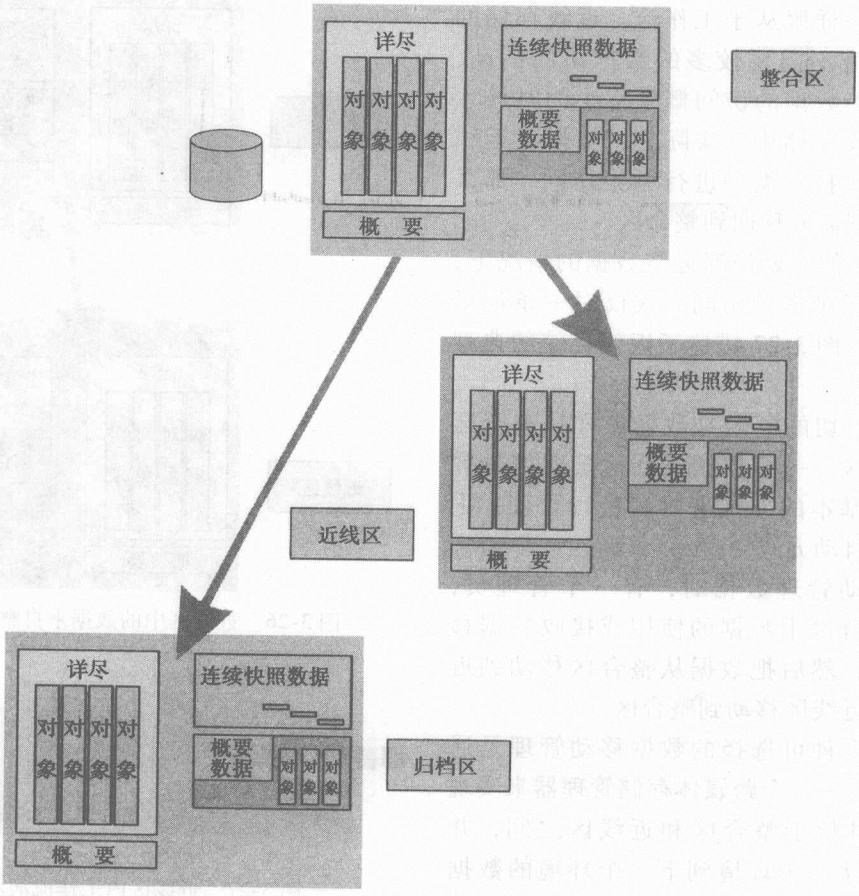


图3-25 数据离开整合区后的流向

3.3 近线区

近线区是整合区的一种缓存形式。近线区可能用于缓存也可能不用于缓存，这完全取决于数据仓库的性质。当数据仓库的整合区很大时，通过近线区缓存数据来降低整合区的工作量就很有意义。当整合区中的数据不是很多时，使用近线区可能就不是很必要。

近线区的使用出于两个原因——成本和性能。近线存储成本大大小于磁盘存储。因此，如果不能负担得起用于整合区的磁盘存储硬件的昂贵成本，那么整合区中的大量数据可以被下载到近线区中用以削减大量的成本。

通过将访问率较低的数据下载到近线区可以大大提升性能。因为只有将那些很少访问的数据送入近线区，整合区磁盘存储器中保留的数据才能避免由“意外”的大量不准备使用的数据所带来的开销。

近线存储是将数据连续地存储在自动管理的磁盘上。近线存储用于大量数据的廉价存储。数据在存入近线存储器后仍然可以通过电子方式获取，但存储代价相较于将整合区的全部数据存入磁盘存储器明显减少。

图 3-26 描述了来自于整合区的近线区数据。

数据被置于近线存储后，它将像任何其他环境一样服从于工作流。近线存储的典型工作流不需要较多的数据访问活动，原因是仅当数据的访问概率很小的时候才被放入近线存储中。实际上，如果对近线数据的某些特定类型进行频繁访问，那么这部分数据需要移回到整合区。

在极少的需要访问近线数据的情况下，或对一串记录进行访问，或仅对一条记录进行访问。图 3-27 描述了近线存储的典型工作流。

一个急切的问题是数据怎样从整合区进入近线区。一般数据在到达其所处位置后有两种基本的方式来对其进行管理：手动方式和自动方式。

当手动管理数据时，有一个管理员，他监督整合区中数据的使用或接收数据移动的请求，然后把数据从整合区移动到近线区或从近线区移动到整合区。

另外一种可选择的数据移动管理是通过 CMSM——一个跨媒体存储管理器来实现的。CMSM 位于整合区和近线区之间，并自动管理从一个环境到下一个环境的数据移动。

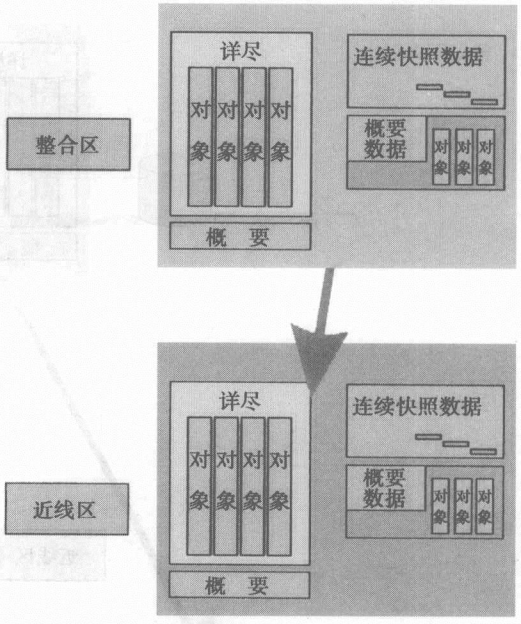


图 3-26 近线区中的数据来自整合区

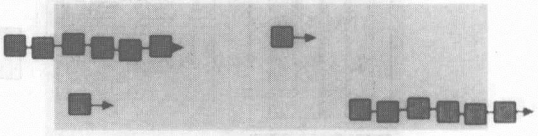


图 3-27 近线区内工作流演示

CMSM可以一种透明性模式运行。透明模式下的CMSM检查进入系统的请求，查看是否有查询近线存储管理数据的请求。当一个需要查询近线存储管理数据的请求到达，CMSM将请求事务排队，并去查找所请求的数据，再把数据下载到磁盘存储上，然后将事务出队并执行事务。开始执行后，事务就能够找到所有它所需要的数据，这些数据由CMSM放到磁盘中。

CMSM还有一些其他的操作方式，上面所述的只是多种操作方式中的一种。

图3-28显示了整合区与近线区之间的接口。

通常，近线区中的数据是整合区中数据结构和格式的镜像。近线区中数据的设计、DBMS以及DBMS的发布与整合区中相应的模块是一致的。整合环境和近线环境的数据之所以极端相似，一个非常重要的原因是数据在两个环境中需要有效地交换。显而易见，数据需要从整合环境移动到近线环境，但再从近线环境移回到整合环境就不是很常见了，只有对数据的访问概率上升时才把数据从近线环境移回整合环境。因此当数据的格式、结构、技术一致的时候，从近线环境移动到整合环境很容易，但若缺少其中任何一项时，这种移动都会变得相当困难。

图3-29给出了近线环境和整合环境中结构、格式及技术都一致的数据。

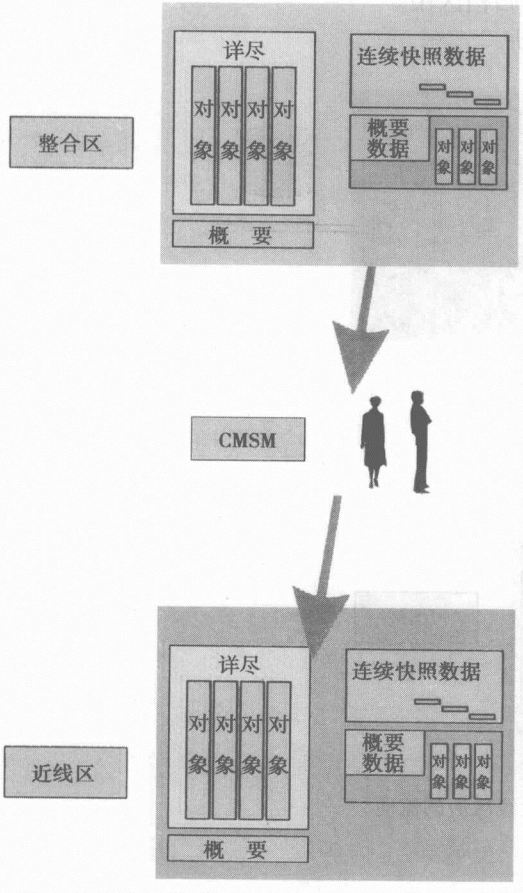


图3-28 管理整合区和近线区之间数据流的两种必要方式

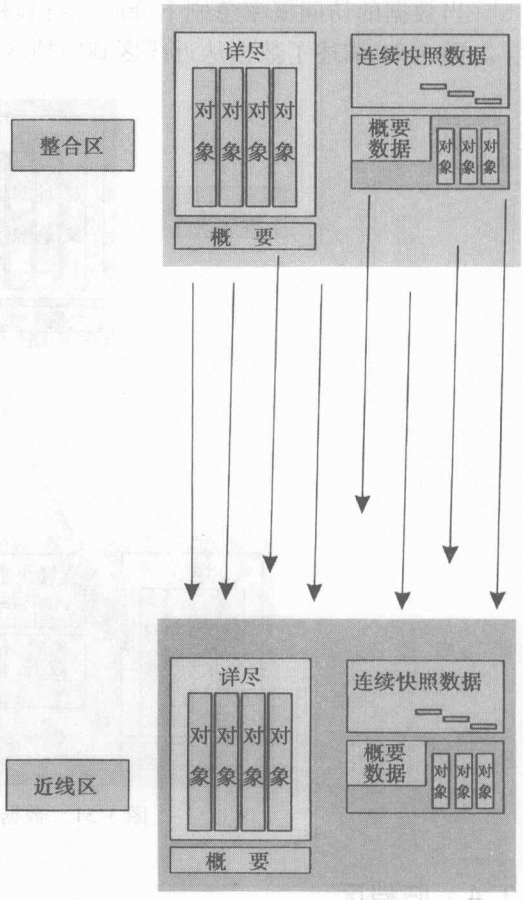


图3-29 结构上，近线环境是整合环境的一个拷贝

近线环境的一个主要优势是它能管理超大容量的数据，远远超过交互环境和整合环境，在近线环境中，管理几百 TB 的数据都是可能的。

图 3-30 显示了近线环境管理大量数据的能力。

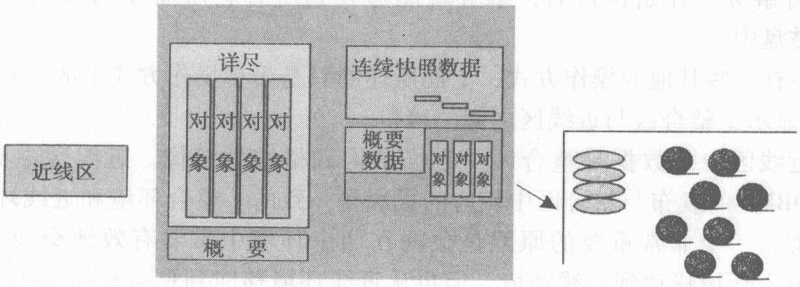


图 3-30 近线环境可以管理巨大容量的数据

离开近线区以后，数据一般进入归档区（Archival）。值得注意的是，归档区中的数据可能直接从整合区中得来而不经过近线区。然而，如果数据已经进入近线区，那么一般情况下就会从近线区进入归档区。

当数据的访问概率急剧下降时就将其移动到归档区中。

图 3-31 描述了数据从近线区到归档区的移动。

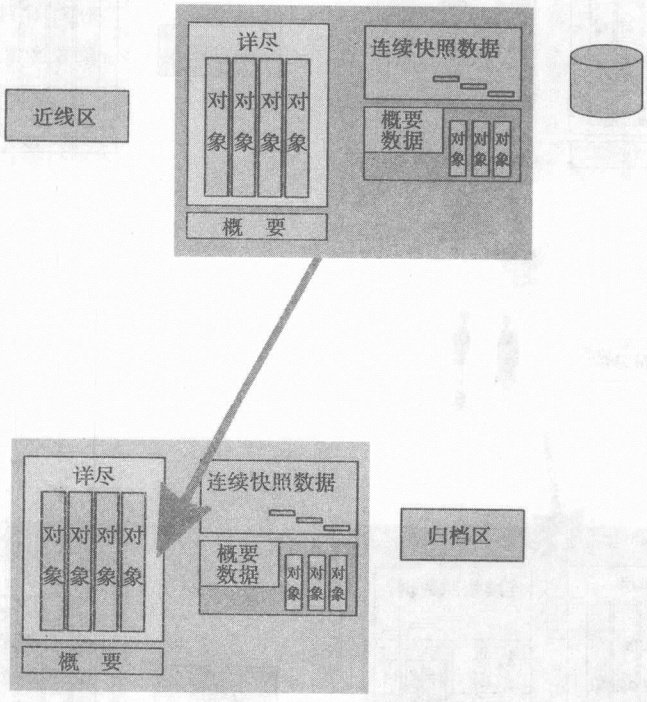


图 3-31 数据离开近线区后的流向

3.4 归档区

归档区是用来存放访问概率变得非常小的数据的区域。有时候，数据被存放在归档区

并非因其被访问概率，而是由于一些法律的原因，因为有时数据的存储是由政府长期授权的。

过去，将数据进行归档常常是一条单行道，进入档案的数据变得不可访问。当工作人员去打开一盒用于归档的磁带时，常常会从塑料容器上掉下氧化物碎屑来，这当然意味着磁带已经分解，也就意味着因为无法读取，磁带变得一无是处。

如今当数据被存放在归档存储中时，这些数据必须在未来的某个时间点是可读取的，否则归档环境就是一种对时间和金钱的浪费。

从许多方面来说，构建一个归档环境就像构造一个时代文物密藏器。人们把各种各样的东西存放在时代文物密藏器中，谁也不知道这个密藏器会在什么时候被谁打开。这个比喻同样也可以用于存放在归档环境中的数据上。归档环境填满数据后，未来需要数据的用户是未知的，需要数据的时刻也是未知的。

因此，归档环境需要组织数据，以便数据在“时代文物密藏器”中能够完全自给自足。

图 3-32 概括了数据如何从整合环境或者近线环境进入归档环境。

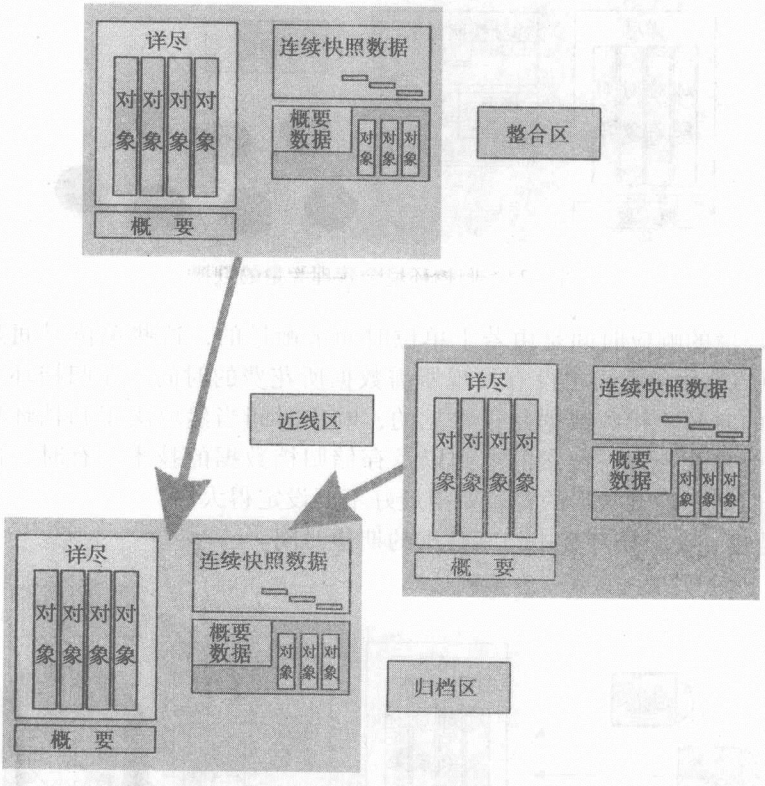


图 3-32 归档区的数据可以来自整合区或近线区

与归档环境相关的工作流是很不同寻常的。很长一段时间——数月甚至数年——常常对于归档数据没有任何访问，然后有一天对于数据有了需求——或者是几条记录，或者是一个长的完整的连续数据串。

对于归档数据来说最大的问题通常是如何找到需要的数据。其他环境的重点在于保证

亚秒级的响应时间，而对于归档数据来说问题却是能否找到需要的数据。通常有大量的归档数据，并且搜索数据的基本准则是模糊不清的，于是在归档环境中查找数据就像在干草堆中寻找谚语中说的那根针一样困难。

图 3-33 描述了与归档环境相关的工作流。

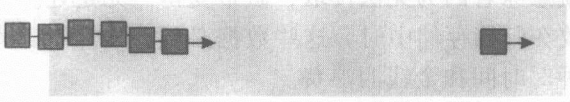


图 3-33 工作流进入归档区演示

归档区的数据量是巨大的。随着时间的流逝，人们希望在归档区中存储比其他任何地方都多的数据。在数据仓库生命周期的初期，档案中存放的数据量通常是很小的。但是随着时间的推移，当数据仓库变得成熟时，它的归档数据会累积、发展，进而包含海量数据。

图 3-34 展示了能在归档区中找到的大量数据。

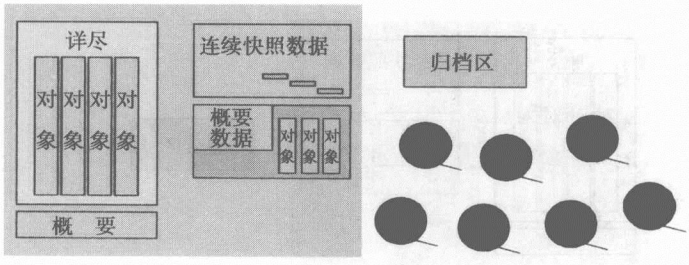


图 3-34 归档环境会管理大量的数据

访问归档环境的响应时间是由若干单位时间来衡量的，这些单位时间是指截止到在 DW2.0 结构中的其他位置再也没有发现所需数据所花费的时间。在归档环境中预期花费几天甚至几周的时间才找到数据是很常见的，响应时间当然取决于归档环境中的数据量大小、数据索引是否合理、搜索的准则以及存储归档数据的技术。有时一次搜索也许非常快，但是我们对于检索数据的普遍期望最好不要设定得太高。

图 3-35 描述了在归档环境中检索数据的期望时间。

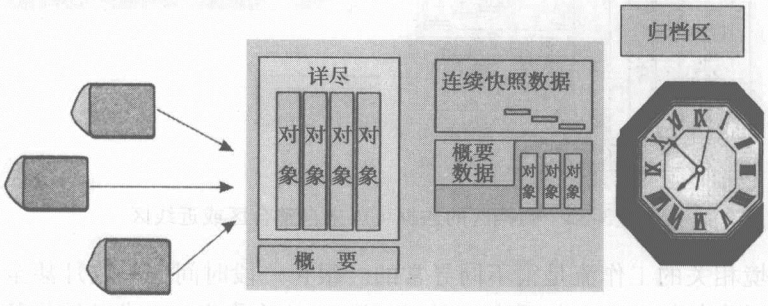


图 3-35 访问归档环境是以小时甚至是天来衡量的

有时，当完成一次搜索时，数据可能会从归档环境移动到整合区。这种归档数据的复原表明我们有理由怀疑这时大量分析和访问需要数据。在大多数情况下，进入归档环境实在是一段痛苦的经历。通过把使用频繁的归档数据移回整合区可以缓解不得不再次进入归档环境进行检索所带来的经常性痛苦。

图3-36展示了数据可以从归档区移入整合区进行重新存储。

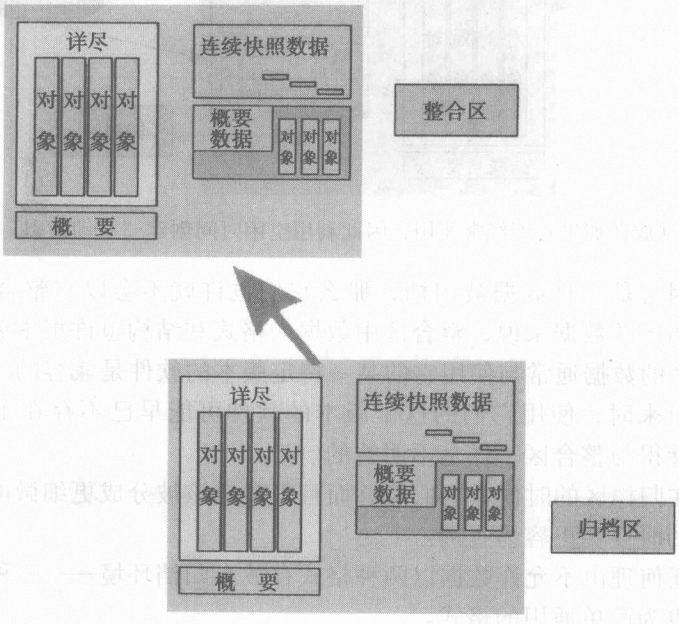


图3-36 有时数据会从归档环境移入整合环境

使用归档数据所能做的最有用的事之一就是创建被动索引。归档环境通常只是一动不动地停在那里，偶尔会有些人往归档环境加载更多数据，但是绝大多数情况下，归档环境基本上是缺乏活力的。

然后有一天，某个人需要归档环境里的数据，系统会突然通过大量的数据。在归档环境中，会出现一段特有的长时间的静止，紧随其后的是短时间的剧烈的大量活动。就算在归档环境中找到数据，这次搜索也始终是不太可靠的。

这些情况下应该创建和使用被动索引。

在通常环境下，索引是用来使用已知的要求对数据进行快速访问的。但是对于归档数据来说，几乎没有可预知的访问路径。因此，只有当归档区的数据只是呆在那里的时候，基于可能的访问路径来创建索引才是很好地利用时间。创建好被动索引后，搜索归档区时凭借一点运气，要寻找的数据能够快速容易地找到。只有在所查找的数据还没有建立被动索引时，才需要对数据进行完整的顺序搜索。

图3-37描绘了为可能到达和通过归档区的路径创建被动索引。

当数据被送往归档区时，数据在整合环境或近线环境所具有的结构是否能够被适当地保持下来是不确定的。保持或者不保持数据结构都有各自的优缺点。保持经过归档区的数据的结构的一个优点是实现起来很容易。数据简单地以一种格式读入，然后以同样的格式写出。这大概就像获取数据一样容易。但也有一些原因使得这种方法也许不是最佳

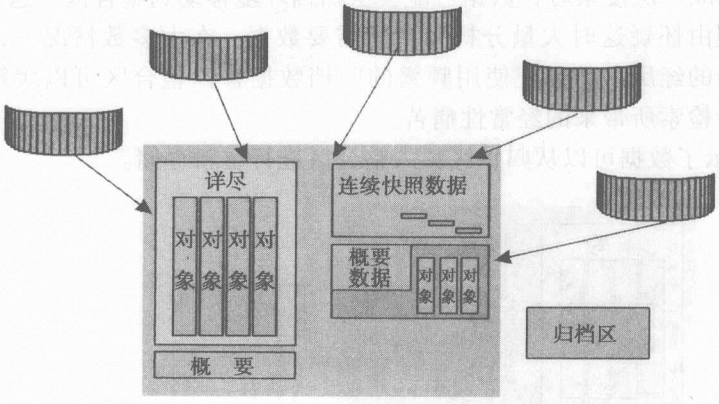


图 3-37 归档数据放在那里并不经常使用，因此利用空闲时间创建“被动索引”是个不错的主意

的，其中一个原因就是一旦数据被归档，那么它们也许就不会以与整合环境相同的方式被使用。对于归档区的数据来说，整合区中数据的格式和结构也许根本就不合适。

另外，整合区的数据通常与使用它的某一特定版本的软件是兼容的。到归档数据被从归档环境中检索出来时，使用它们的这个版本的软件可能早已不存在了。因此，把归档区的数据结构设计得与整合区一样是不明智的。

当数据存放在归档区的时候，它们能够而且常常应该被分成更细微的部分。这样当搜索或者访问它们的时候会更容易查询。

另外，没有任何理由不允许数据以两种格式存放在归档环境——一种是整合区中的源格式，另一种是更为简单通用的格式。

图 3-38 描述了归档区中数据的双重结构。

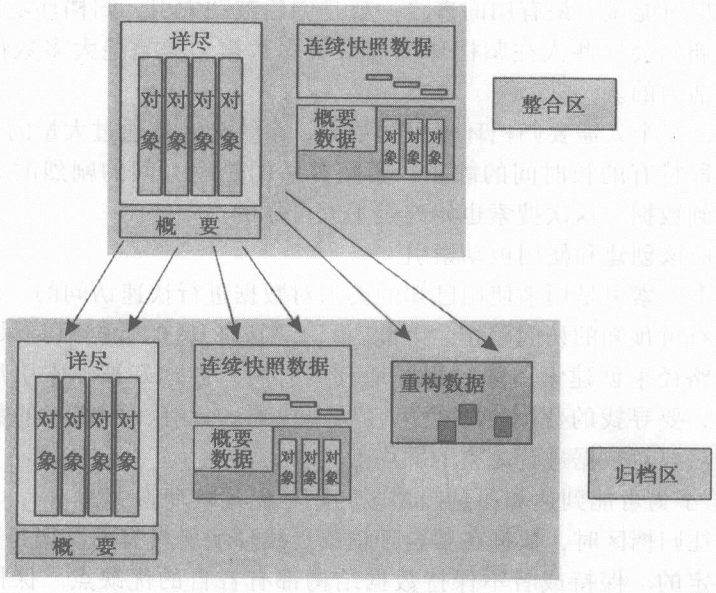


图 3-38 当数据被送入归档环境，数据的结构也许会被保留下来，数据也许会被重构，或者两者都发生

归档区中的数据需要尽可能避免受软件版本的限制和约束。

如图 3-39 所描述的那样，对于从归档区中找到的数据，有一种可以预知的访问模式。

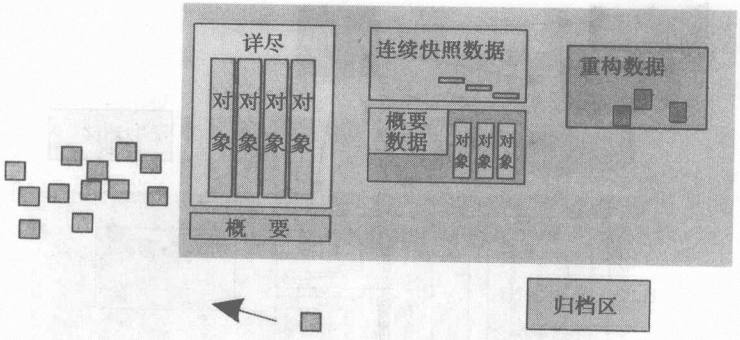


图 3-39 归档环境的访问模式：a）非常不频繁的访问；b）大多数访问都是针对大量的顺序数据的；c）只有极少的针对特定数据的访问

归档区的数据很少被访问，而且访问的时候，通常情况下整个归档数据组都会被访问到，检索归档环境中的单条记录的情况极为罕见。

访问归档环境中的数据很有趣的一方面是，通常，数据需要基于模糊的字段或数据值来定位。偶尔会出现以“标准”码和标识符访问数据的需求，但经常有基于非常不正规的数据类型的访问。

如图 3-40 所描述的那样，由于归档数据的数量和归档数据需要长时间保留的事实，归档区并不具备引用完整性约束。

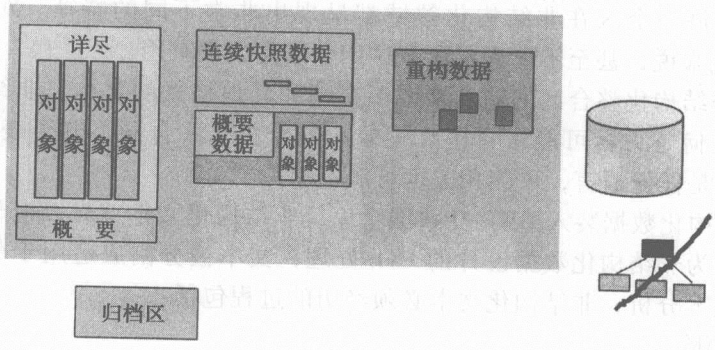


图 3-40 通常情况下不强制要求归档区具有引用完整性

人们常常以查找任意相关数据并将其移动到整合区或探索程序中为目的来搜索归档区。但是有时归档区进行自搜索也是很有意义的。换句话说，归档区可以被当做决策的依据来使用。然而，这种方法的缺点包括但是不局限于以下几点：

- 在归档区中确实有大量的数据。
- 归档区中的数据需要被顺序搜索。
- 没有为待完成的搜索提供有用的索引。

进一步说，与其他区域相比，可供归档区使用的数据查询和分析技术很有限。

但是，有时可以通过一种与操作系统无关的方法访问和分析归档区。图 3-41 描述了这种能力。

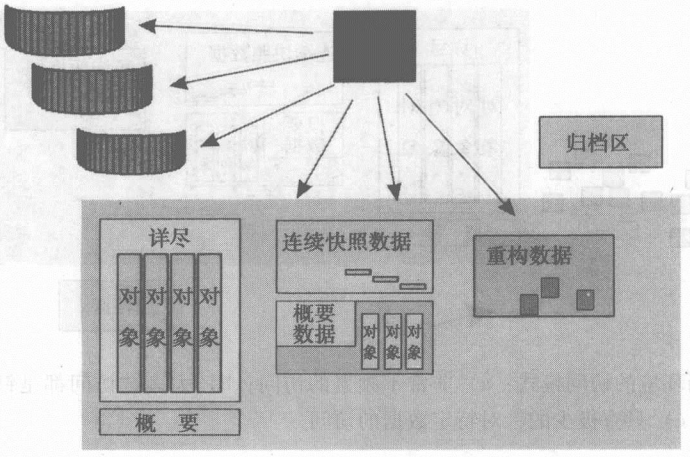


图 3-41 归档环境始终可能对自己进行顺序搜索和处理

3.5 非结构化处理

DW2.0 环境主要有两个方面或领域。一半的 DW2.0 数据结构格式是为结构化的数据设计的，到目前为止它也是这一章讨论的主题。DW2.0 环境的另一半是非结构化数据的领域。

虽然 DW2.0 的四个分区都适用于 DW2.0 环境的非结构化数据，但是与 DW2.0 的结构化方面相比，每一个区在非结构化领域都呈现出非常不同的特征。对于非结构化的 DW2.0 数据领域来说，甚至不能确定所有这四个区是否都有用。

DW2.0 的非结构化整合区的输入来源于文档和其他格式的文本。非结构化数据输入可能来自几乎任何地方：可能来自医疗记录、安全报告、合同、电子表格、检验报告，等等。文本可以是任何语言，可能相关也可能不相关。

为了把非结构化数据装入 DW2.0 数据仓库，非结构化文本首先以电子格式聚集在一起，然后经过专为非结构化数据设计的 ETL 处理，文本被分割成适用于分析处理的文本块。为了适合文本分析，非结构化文本必须经历的过程包括：

- 无用词消除
- 分词
- 特殊/通用分析
- 可替换拼写分析
- 分组数据的分类

通过这些严格的过程后，文本就为分析处理做好了准备。

在非结构化整合环境中有几种类型的数据，其中一些如下：

- 内部和外部分类：一个分类就是一组具有相互联系的词汇。非结构化文本环境既包括内部（有时叫做“主题”）创建的分类，也包括可来自几乎任何地方的外部分类。

- 被捕获、被编辑的文本：被捕获、被编辑的文本是指那些通过了非结构化的ETL处理并且被放入数据库——标准关系型数据库的文本。
- 链接：链接是那些联系非结构化数据和结构化数据的数据。
- 简单指针：非结构化的数据文本偶尔会驻留在其他环境中，只有引用它的索引才能进入非结构化的交互数据仓库中。

图3-42描述了通过文本ETL并且进入非结构化整合环境的文本。

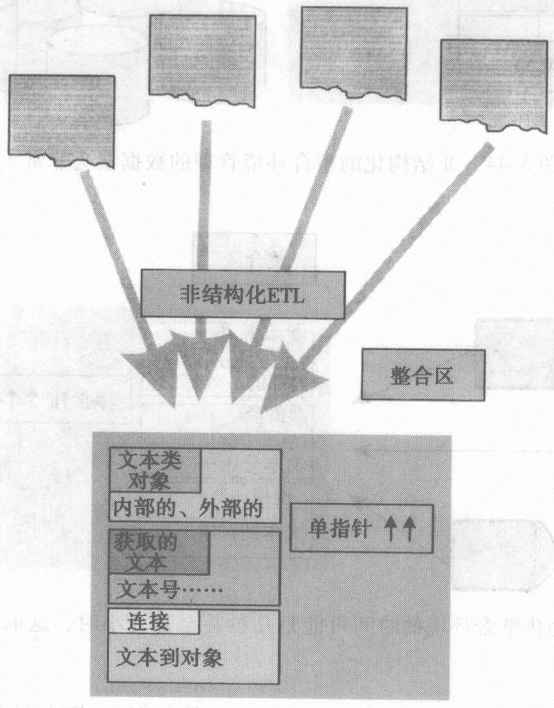


图3-42 非结构化整合层的输入来自哪里

非结构化的整合环境有着与其他每一个DW2.0分区都相似的工作流。图3-43描述了非结构化的整合区工作流的特点。

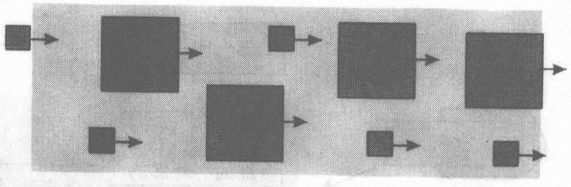


图3-43 非结构化整合区内部正在进行的工作流模拟

图3-44强调了非结构化的整合工作流如何充满了大动作，考虑到文本环境的特点，这就不足为奇了。然而，因为文本数据库的规模变化范围较大，所以在这个环境中也会存在一些小规模的活动。

因为非结构化整合环境的工作流较为复杂，所以响应时间的期望值也很复杂。图3-45显示了非结构化整合环境中的响应时间。

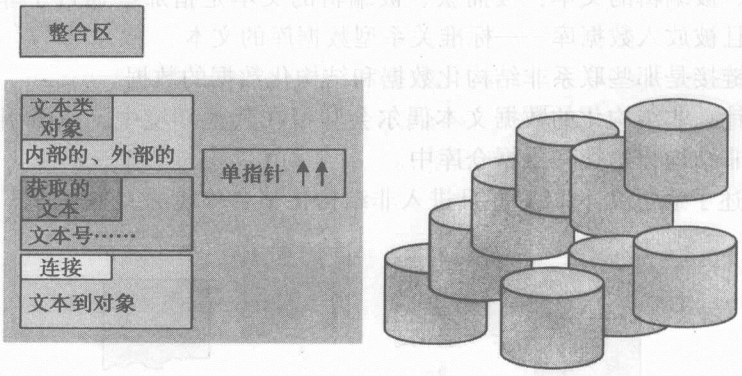


图 3-44 非结构化的整合环境管理的数据量会非常大

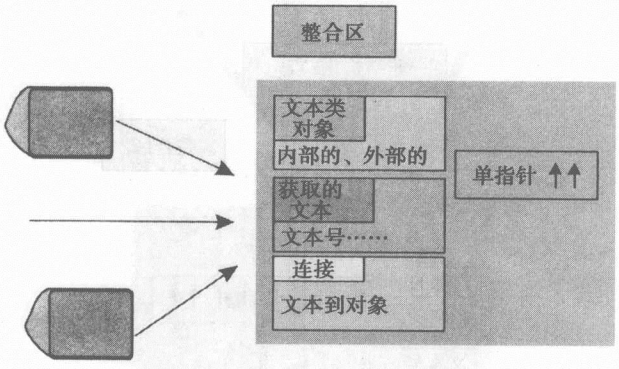


图 3-45 访问非结构化整合环境的时间可能是几秒甚至是几小时，这取决于事务提交时刻正在处理的工作量

在非结构化整合环境中基本上有两种活动——数据的加载和数据的访问。非结构化的文本数据几乎不可能更新。当一个文本描述或工作被写入后，如果需要修改，那么只能重新写入。因此，逐渐地或者部分地更新文本数据显然不现实。

图 3-46 描述了非结构化整合环境的两种基本活动。

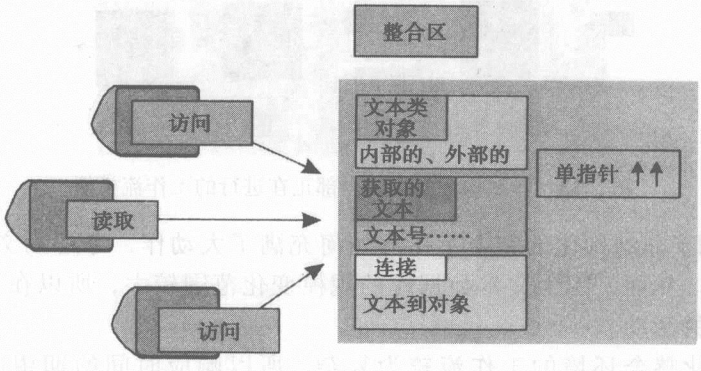


图 3-46 在非结构化整合区中，事务可以访问或者读取数据

在DW2.0中非结构化环境与结构化环境很不相同。通常只有一个非结构化的整合区，而是否需要一个非结构化的近线区还是个疑问。

然而，有时还是会为了非结构化数据而使用归档环境。当数据的访问概率降低时，就会被放入非结构化的归档区中。图3-47描述了非结构化文本数据向非结构化归档环境的移动。

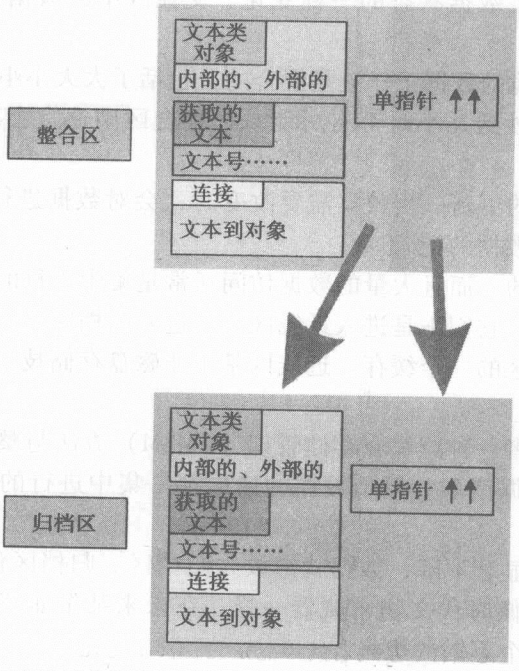


图3-47 非结构化的整合数据偶尔需要被放入非结构化归档环境

3.6 企业用户的观点

将数据分为不同的部分对于企业用户来说是很普通、很自然的。将构架展示给终端用户，使其看起来像是只有一个单一的数据存储并不是一个好策略。终端用户知道数据是以时间为依据存放在不同地方的。

另外，终端用户知道可以在DW2.0中找到文本的、非结构化的数据，并且可以在这些数据中混入查询语句。而且，终端用户知道有一种设施——元数据存在，元数据让终端用户能在环境中游刃有余。

最后，终端用户知道不同种类的处理一般会出现在不同的地方。如果终端用户想要进行在线更新和分析，交互区可以实现。如果终端用户想要进行整合的分析，则可由整合区来完成。如果终端用户想要查看足够陈旧的数据，那么归档区可以实现。

唯一对终端用户透明的区域是近线区或者叫做数据仓库实用工具区。即便那样，终端用户还是希望有机会访问和分析近线区中的数据。

3.7 总结

一般情况下，数据从交互区进入DW2.0环境。数据可以通过ETL或者直接进入

DW2.0 环境。交互区是一个面向应用的区域，这个区域可以进行数据更新，并且支持 2～3 秒的响应时间。交互区中的工作流小而快，不允许大的事务通过。

数据在交互区中以一种随机、快速而且少量的模式被访问。交互区中只有有限的历史数据。

整合区数据在进入该环境之前就已经经过整合了。通常，数据整合的工作是由 ETL 工具完成的。整合代表了数据状态的一种变化。交互区中的数据是面向应用的，而整合区中的则是企业数据。

进入和离开整合环境的数据工作流是混合的，包括了大大小小的事务。整合区的响应时间也是混合的，从几秒到几小时都有可能。在整合区中通常会有大量的数据存在，数据一般为 3～5 年。

整合区中没有数据的更新。当数据需要改变时，会对数据进行快照然后插入数据库，同时也会创建一条历史数据的记录。

访问是不经常发生的，而且大量的数据访问通常是集中进行的。

当数据离开整合区，它们不是进入近线区就是进入归档区。

近线区就像是整合区的一个缓存。近线区基于非磁盘存储技术运行，其中包含了整合区中数据的镜像。

近线区通过人工或者一种跨媒介存储管理（CMSM）方法与整合区连接。近线区的工作流主要是不频繁的数据读取。但是数据的读取都是集中进行的。当数据的访问概率下降时就被放入近线区。

当数据的访问概率显著下降，数据就被放入归档区。归档区包含了自主式的数据包。这些自主式的数据包就像时代文物密藏器一样，在未来某个非特定时间被打开。为归档数据创建被动索引是一个不错的想法。

通常归档环境中有大量的数据，其中会有上百年的数据也是可以理解的。为了实用，归档数据必须和软件版本以及产品约束无关，这是因为当需要数据的时候，相同版本的产品不太可能继续使用。

非结构化的数据只有先被整合后才能对文本分析有用。在进入非结构化的 DW2.0 环境前非结构化数据必须通过 ETL 层。

在非结构化的环境中通常会有大量的数据。对于非结构化数据来说可能没有归档区和近线区。

第 4 章　DW2.0 中的元数据

DW 构架中必不可少的组成部分之一是元数据。第一代数据仓库中不提供或是后来才想到使用元数据，而在 DW2.0 中，元数据成为数据仓库的基石。

很多原因使元数据变得如此重要。首先，元数据对开发者来说很重要，他们必须将自己的努力与之前所做的工作联系起来。第二，元数据对技术维护员来说也很重要，因为他们必须处理日常问题以确保数据仓库有序工作。元数据对于对终端用户来说可能是最重要的，因为终端用户需要找出都有哪些可能可用于新的分析。

理解 DW2.0 中元数据重要性的最好方法是将其看作大型公共图书馆中的卡片目录。在这样一个大型公共图书馆中信息是怎样被搜索到的呢？人们是否进入图书馆以后逐排寻找自己想找的书呢？这当然可以，但会浪费很多时间。比较合理的做法是直接查询图书卡片目录。相较于遍历图书馆中每本书的手工查找方法，查找卡片目录的方法大大提高了查找效率。

当目录中存在读者想要寻找的书时，读者便可以直接去指定地点找到该书，这样做大大节省了查找资料的时间。

DW2.0 中元数据也扮演着类似于图书馆中的目录卡片一样的重要角色。元数据允许分析人员查看其组织结构，并掌握已经完成了什么分析。

4.1　数据和分析的可复用性

下面我们讨论终端用户。终端用户置身事外但对信息存在需求。这种对信息的需求可能源于管理指令，也可能源于企业委托，或者纯粹是终端用户出于个人的好奇。不管需求源自何处，终端用户都在想办法得到这些分析数据，而元数据就成为其求助的对象。元数据使分析人员能够确定哪些信息是可用的。一旦分析人员确定了数据最可能的来源，便可以开始访问这些数据。

没有元数据，分析人员很难识别数据的可能来源。分析人员可能熟悉其中一些数据的来源，但并不一定知道所有数据的来源。在这种情况下，元数据的存在省去了我们很多不必要的工作。

同样，终端用户也可以利用元数据来判断是否已经完成某一分析，回答问题就像察看他人做了什么一样简单。如果没有元数据，终端分析员将永远不会知道哪些工作已经完成。

鉴于以上这些原因（其实还有更多的原因），元数据成为 DW2.0 构架中非常重要的一部分。

4.2　DW2.0 中的元数据

元数据在 DW2.0 中起着特殊的作用。DW2.0 中的每个区中都有各自的元数据，其中包括交互区元数据、整合区元数据、近线区元数据以及归档区元数据。

归档区元数据不同于其他元数据，这是因为归档区元数据直接置于归档数据中，以确保元数据不会跟其所描述的基础数据分离或丢失。

图 4-1 描述了 DW2.0 构架中元数据通常所处的位置。

DW2.0 中有通用的元数据构架，实际上，有两种并行的元数据构架——一种用于非结构化环境中，另一种用于结构化环境中。图 4-2 描绘了 DW2.0 元数据结构层次的高层。

对于非结构化数据而言，有两种类型的元数据——企业型和本地型。企业元数据被认为是通用元数据，本地元数据被认为是专项元数据。

对于结构化元数据来说，有三个级别——企业级、本地级、业务或技术级。这些不同类型的元数据之间有着非常重要的联系，解释这种关系的捷径是从本地级开始讨论。

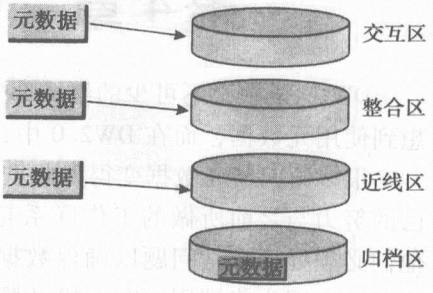


图 4-1 元数据及 DW2.0 环境

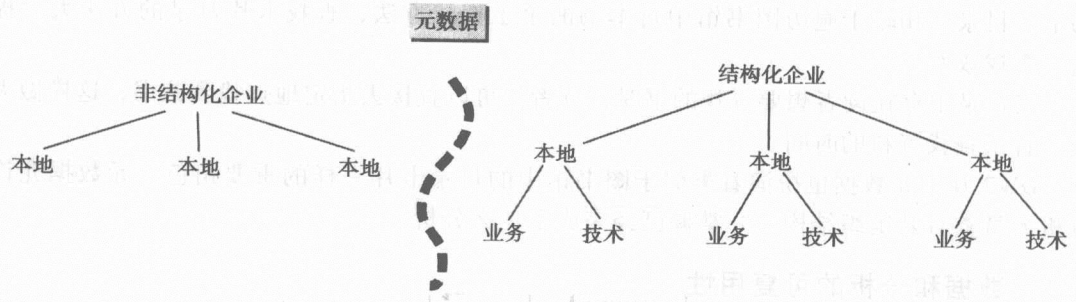


图 4-2 元数据的总体构架

本地元数据之所以是最佳的研究起点是因为它是很多人最熟悉的元数据类型。本地元数据存在于很多不同的表格和位置中。它存在于 ETL 处理、DBMS 字典以及商业智能领域中。

本地元数据是这样一种元数据类型，它存在于一种工具中，而这种工具对描述和其直接相关的元数据非常有用。例如，ETL 元数据涉及数据源、目标以及数据从源地址送到目的地址时的数据转换等问题。DBMS 字典元数据与表格、属性及索引有关。商业智能领域的元数据则是用以针对那些用于分析处理的数据。除此之外，还有很多类型的本地元数据。

图 4-3 举例说明了本地元数据。

本地元数据的存储于对利用本地元数据非常重要的一种工具或一项技术当中。另一方面，企业元数据则储存在对 DW2.0 环境下的所有工具和过程来说都很重要的本地当中。

图 4-4 说明企业元数据存储在 DW2.0 环境中的各个区。

从图 4-4 中可以看出，DW2.0 中各区之上是企业级元数据的集合，所有的企业元数据一起形成一个知识库。实际上，除了归档区之外的所有区都将它们的元数据存储在知识库中。

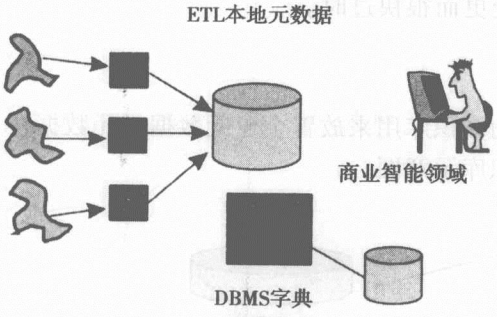


图 4-3 本地元数据存在于很多地方

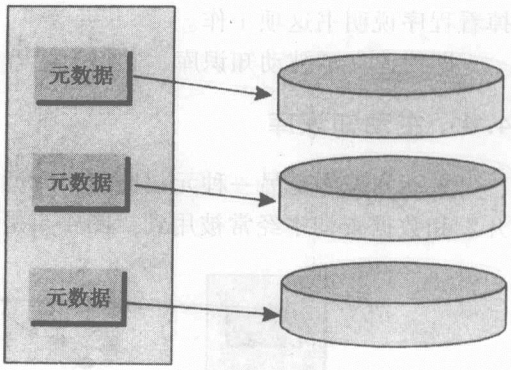


图 4-4 元数据存储在知识库中

4.3 主动知识库/被动知识库

主动知识库和被动知识库是两种基本类型的元数据知识库。主动知识库里面的元数据随着系统的发展和查询活动变化而不断地进行交互。被动知识库里的元数据不能直接随系统的发展和/或终端用户的查询进行交互。

图 4-5 描述了被动知识库。

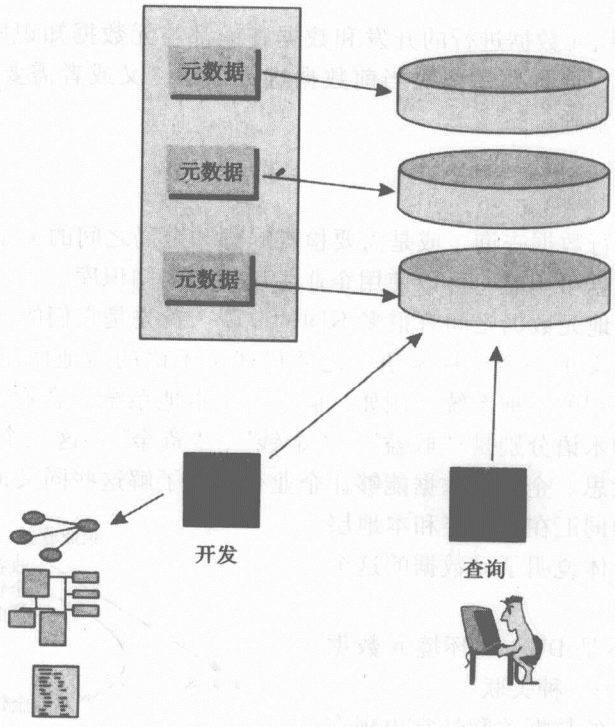


图 4-5 被动知识库

不推荐使用被动知识库，因为终端用户和开发者的活动是独立于元数据知识库的。因为大部分的机构都会尽可能地减少工作量，降低开支并尽快完成任务，所以任何可选择的工作都将无法完成。被动元数据知识库如同程序说明书一样，经验丰富的开发者会省

掉看程序说明书这项工作。

即使建立了被动知识库，它也会随着系统的变更而很快过时。

4.4 主动知识库

主动知识库是另一种元数据知识库类型。主动知识库用来放置企业元数据，元数据在开发和数据查询中经常被用到。图 4-6 是主动知识库示意图。

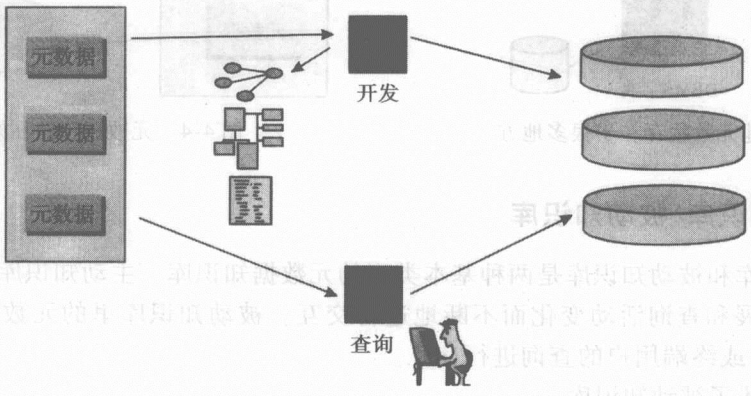


图 4-6 主动知识库

图 4-6 说明，使用元数据进行的开发和数据查询都与元数据知识库密不可分。例如，当数据模型出现变化，或者是需要对当前数据进行描绘，又或者需要源数据时，都依赖于主动知识库作为其基础。

4.5 企业元数据

当需要对表格进行数据查询，或是需要检查局部和全局之间的关系，再或是分析员仅仅希望知道有哪些数据可用时，可以使用企业主动元数据知识库。

企业元数据与本地元数据之间有很多不同的关联。首先是它们的语义关联。在语义关联中，企业为公司定义了一个全局术语，之后描述了术语的本地使用，并用指针指向可以在其中找到术语使用的本地系统。例如，假设三个本地系统与企业术语“收益”相关。这三个本地系统中的术语分别是“收益”、“金钱”、“资金”。这三个不同的词汇在这个例子中代表相同的意思。企业元数据能够让企业很好地了解这些同义词术语。

元数据使得各种词汇在企业层和本地层得以统一，图 4-7 具体说明了元数据的这个特性。

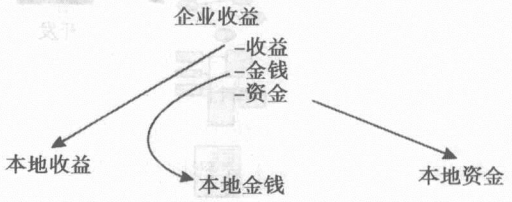


图 4-7 企业元数据与本地元素数据关系的例子

但语义关联并不是 DW2.0 环境元数据中企业/本地间的唯一一种关联。

另一种非常重要的数据关联经常出现在企业对象域定义中。图 4-8 说明了这种定义方式。

该图给出了一个主要的对象域“顾客”，其定义在企业层。在本地层可以找到关于顾客的不同信息。在第一个本地系统中存有客户的姓名及地址信息，第二个本地系统中存

有关于顾客年龄和购买偏好的信息，第三个本地系统掌握顾客收入、学历、社会保险号。

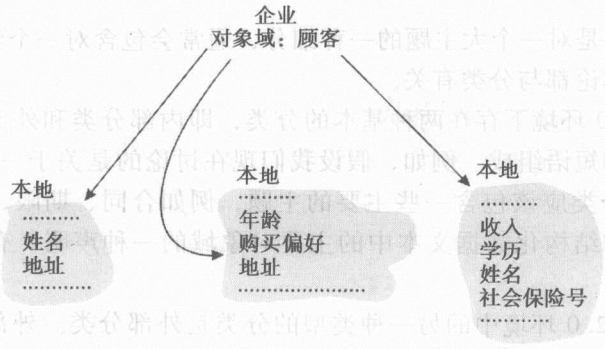


图 4-8 在企业层关于对象域的实例说明

企业元数据层可用于确定本地系统在哪里存储了对主要业务对象的支持数据。

4.6 元数据和记录系统

元数据同样可以用于为企业的数据对象和数据属性定义记录系统。在记录系统中，每个数据的最终来源是明确的。图 4-9 对这点进行了说明。对于企业中的主要对象的多种数据属性而言，有多个记录系统是正常的。

值得注意的是，在前面三个例子中，数据定义和本地元数据层与企业元数据层间的关联定义之间存在重叠的部分，但它们之间也有所差异。这些关系间的差别是十分微妙的，在 DW2.0 环境下，企业元数据和对应的本地元数据能够体现这些细微的差别。

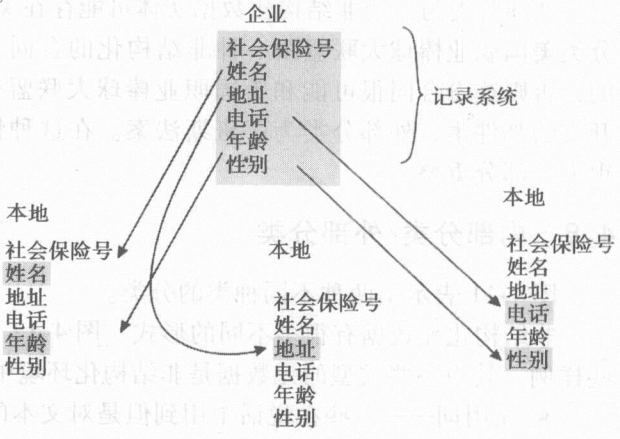


图 4-9 元数据企业层描述的记录系统

在 DW2.0 中还有一种元数据关系同样非常重要。例如，在元数据本地层有两种明显不同的元数据类型，它们是业务元数据和技术元数据。

业务元数据用业务人员的行话来说，是指对数据的业务描述。技术元数据是指对数据的技术描述。例如，业务元数据“收益”定义为“用来支付服务或产品的金钱或实物”。再比如，技术元数据可以看成是表格“ABC”中包含属性定义“REV-DESIGNATED PIC 9999.99”。

图 4-10 展示了元数据本地层有其对于业务元数据和技术元数据的细分。

DW2.0 中非结构化数据有属于自己的元数据。非结构化环境的元数据与结构化环境的元数据有很大不同。我们以分类为例说明非结构化环境下的元数据。

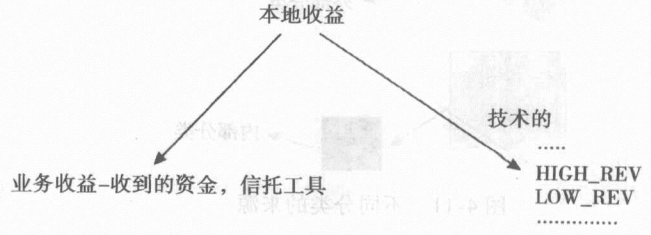


图 4-10 本地业务元数据和本地技术元数据

4.7 分类

简单来说，分类是对一个大主题的一种细分，通常会包含对一个给定主题组件的详细分解。词汇表和本体论都与分类有关。

非结构化 DW2.0 环境下存在两种基本的分类，即内部分类和外部分类。内部分类仅仅由文本中的单词和短语组成。例如，假设我们现在讨论的是关于一系列的合同的问题，对于这一组合同的分类应该包含一些主要的主题，例如合同、期限、协议的长度以及赔偿。内部分类是对非结构化数据文本中的主要对象域的一种声明。有时内部分类被称作文本的“主题”。

在非结构化 DW2.0 环境中的另一种类型的分类是外部分类。外部分类来源很广，外部分类有时就产生于“真实的世界”。外部分类可以包括：

- 萨班斯 - 奥克斯利法案。
- 巴塞尔新资本协议。
- 进出口规定。
- 美国职业棒球大联盟。
- 杜威十进制编码方案。
- Emeril 的食谱。

外部分类与一个非结构化数据实体可能存在关系也可能没有关系。例如，假设将外部分类美国职业棒球大联盟跟一种非结构化的合同文本比较，除非合同是为棒球队员而定的，否则这份合同很可能和美国职业棒球大联盟毫无关联。相反，假设正文是授权公司开支的邮件集，外部分类为萨班斯法案。在这种情况下，分类与非结构化数据实体间有很大一部分重叠。

4.8 内部分类/外部分类

图 4-11 表示了两种不同种类的分类。

非结构化元数据有很多不同的形式。图 4-12 给出了不同类型的非结构化元数据的一些样例。其中一些类型的元数据是非结构化环境中比较普遍的，它们包括：

- 无用词——一些在说话中用到但是对文本的意义不重要的词。典型的无用词包括：一个、和、是、那个、哪个、哪里、到。

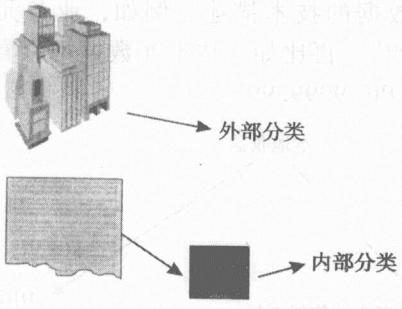


图 4-11 不同分类的来源

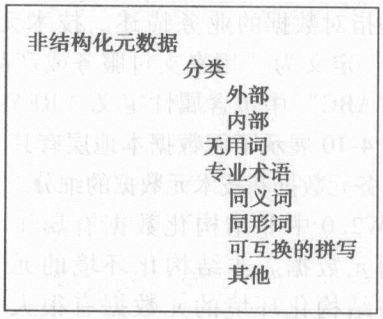


图 4-12 非结构化元数据的一些类型

- 同义词——意思一样但是拼写不同的词。例如 fur 和 coat 这两个单词。
- 同形词——拼写相同但意思不一样的词。例如，“bow of a ship” 中的 bow 和 “bow and arrow” 中的 bow。
- 可互换的拼写——同一个单词的多种可以接受的拼写。例如 color 和 colour。

4.9 归档区元数据

在 DW2.0 环境下，当涉及归档区元数据时，会出现一种异常。在归档区中，与归档过程相关的元数据存储在归档数据本身中。之所以将它们存在一起，是因为假设如果将元数据与其相关的归档数据并排相放，那么它将随时间而丢失。当然，在归档环境中也可以存储独立的元数据集。但是对历史数据的查询最频繁并且也最可能有用的第一存储地点是归档数据本身。

图 4-13 显示了归档数据中应该包含属于它自己的元数据。

图 4-13 归档区中的元数据

4.10 维护元数据

元数据面临的一个重大挑战不是元数据环境最初始的创建，而是对元数据环境的持续维护。改变是永恒的，而改变对元数据的影响跟对其他所有事物的影响一样。

图 4-14 显示了一个发生变化的情况，并表明主动元数据环境比被动环境更易适应改变。

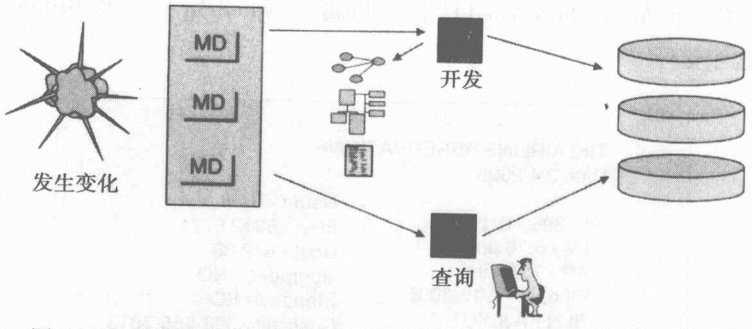


图 4-14 当发生变化时，在主动知识库中进行维护相对比较容易

在被动元数据环境下，很容易忽视改变。一个变化发生以及由此导致的一系列改变在被动元数据环境下会产生延迟。有一天你会忽然发现在元数据环境下长时间无法显示日常正规的变化，由此会导致当你最需要这些数据时，读取到的所谓的最新同步信息实际上没有任何用处。

主动元数据知识库有规律地显示出变化以便对现有系统进行正常的更新和维护。当系统发生变化时，元数据也必然随之改变。

使用元数据与存储及定期更新元数据一样重要。尽管使用元数据有很多方式，但利用终端用户交互式处理的用户接口也许是一种最有效的方法。

4.11 举例说明如何使用元数据

下面举例说明 DW2.0 中元数据的用法。我们考虑以下方案：P. Bruton 是一家航空公司代理商，她要为客户 Bill Inmon 提供带有空乘信息的显示屏。

带有空乘信息的显示屏如图 4-15 所示。

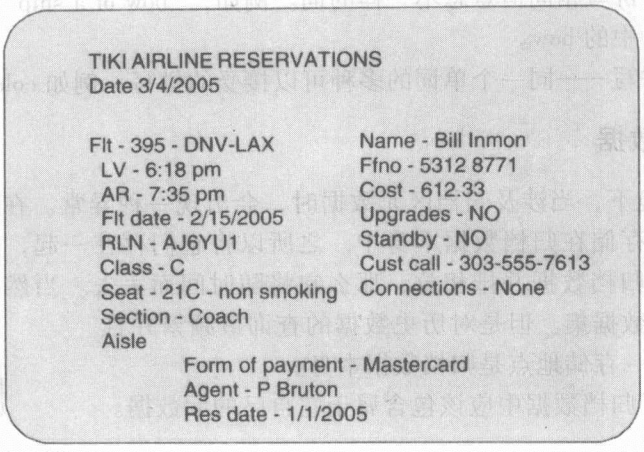


图 4-15 终端用户初始化显示屏或报告

P. Bruton 并不清楚 Tiki 航空公司为她预留的座位。她发现一个让她感兴趣的数据字段，那便是 Tiki 航空公司显示屏上的 CONNECTIONS 字段，如图 4-16 所示。之后，P. Bruton 按下功能键，弹出了一个菜单，如图 4-17 所示。菜单指示让她确定她想要看哪一方面的 CONNECTIONS。选项分别是："别名（AKA）"，"定义（definition）"，"公式（formula）"，"用于何处（where used）"，"归属（owned by）"。P. Bruton 选择了"定义（definition）"这个选项。

图 4-16 用户为自己选择了一个感兴趣的单元

菜单中显示的可用元数据类型如图 4-17 所示。

现在系统开始查询关于 CONNECTIONS 定义的元数据信息。通过对定义的查找，系统显示出关于 CONNECTIONS 定义的相关数据。

在交互模式下元数据定义的显示情况如图 4-18 所示。

需要注意的是，对元数据的访问需要一直使用最初的显示屏。对元数据的显示是基于分析过程之上的。数据交互地显示，并成为分析过程的一部分。

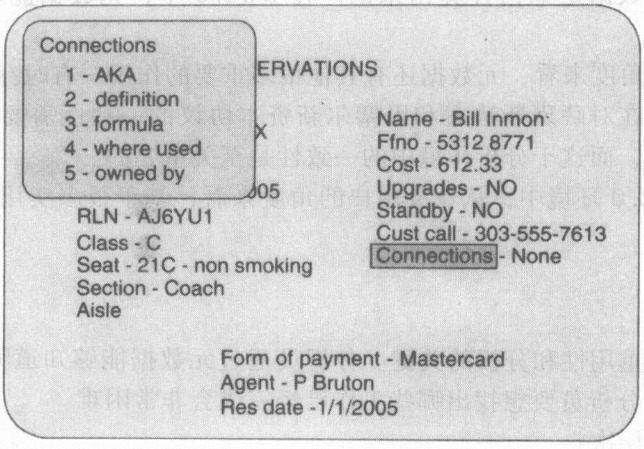


图 4-17 可用的元数据在系统中被显示

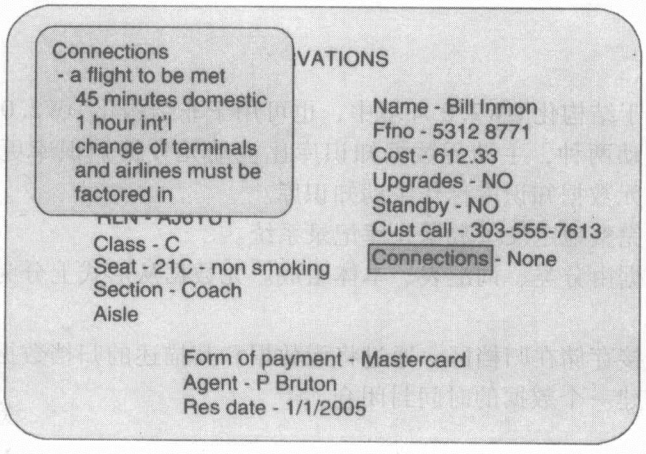


图 4-18 屏幕显示希望获取的元数据

4.12 终端用户的观点

在 DW2.0 中，元数据的用途很广。元数据为不同区的数据提供交互服务，它扮演的角色既像环境的文档，又像为 DW2.0 环境添加数据的线路图。不过它最重要的作用在于为 DW2.0 中的数据内容及关联提供指导。

终端用户对 DW2.0 中的数据和关联需要指导。在 DW2.0 环境下，如果终端用户得到了已存在的那些数据的指导信息，那么就有可能重用这些数据。换一种说法，如果 DW2.0 中不存在元数据，那么终端用户每次创建新的分析都必须从头开始。由于终端用户看不到已经完成的那些分析工作，因此所有工作都必须从头开始。

在多数情况下，每次分析都要重新开始一遍简直是多此一举。运用元数据就不需要这个多余的步骤了。分析员可以在其他分析员的分析基础上进行工作。

从业务用户的角度来看，元数据还有一个很重要的作用，就是可以用于显示数据的继承。在很多情况下，分析员将一个数据单元看作分析工作的一部分，而业务用户需要知

道数据的来源以及数据是如何计算出来的。在 DW2.0 中，元数据能够提供这种重要的功能。

从业务用户的角度来看，元数据还有其他很多重要的作用。有时存在着对数据的一致性需要，例如，存在对萨班斯法案和巴塞尔新资本协议的一致性需要。元数据对审核跟踪提供了关键部分，而这于分析环境中的一致性是至关重要的。

另外，在 DW2.0 环境中，从业务用户的角度来看，基于很多实用原因，元数据有着举足轻重的作用。

4.13 总结

元数据是数据重用性和分析的关键。分析员通过元数据能够知道哪些工作已经完成。如果没有元数据，分析员要想找出哪些工作已经完成会非常困难。

元数据有四个层次：

- 企业
- 本地
- 业务
- 技术

元数据既可用于结构化 DW2.0 环境中，也可用于非结构化 DW2.0 环境中。元数据知识库分为主动和被动两种，主动元数据知识库比被动元数据知识库更有用。在开发和分析阶段交互使用的元数据知识库称作主动知识库。

元数据知识库完整地定义了数据仓库记录系统。

非结构化元数据由分类、词汇表、本体组成。元数据从形式上分为内部元数据和外部元数据。

归档元数据直接存储在归档区。通过将元数据和其描述的归档数据存储在相同的物理存储上，就可以创建一个数据的时间封闭仓。

第5章 DW2.0技术基础设施的流动性

DW2.0作为下一代数据仓库在架构上有很多重要方面——对数据生命周期的认识，包含非结构化数据，以及将元数据作为其基本组成部分。但是对于DW2.0架构而言，面临的最大的一个挑战就是其处于一个所采用的技术可以跟业务同样快速改变的环境中。

因为商业环境总是不断改变，所以从终端用户的角度来说流动性是非常重要的。在某一时刻一个公司需要着眼于利润，这意味着提高价格和扩大销售。而下一年公司又需要着眼于成本，这意味着降低开支和停止扩张。第三年公司可能又需要着眼于新产品和新的收益渠道。每当商业环境发生改变时，就会需要新的类型的信息。而且，由于竞争、科技和经济潮流的改变，对于信息就会不断有新的需求。

如果一个数据仓库建立在难以改变的技术之上，那么企业的这种技术就无法适应商业环境。这意味着，虽然对企业来说，技术同数据仓库本身一样重要，但就其价值而言往往不是最佳的。

在第一代数据仓库中采用了传统的信息处理技术来存储数据。因此，数据仓库就是铁板一块，对于数据仓库很难做出很大的改变。DW2.0认识到并且对上述问题做出相应的处理。

这种情况可以从图5-1看出。图5-1显示了业务需求的永远改变的特性。业务需求的不断改变是不可避免的——就像死亡和税收一样。不同组织间的唯一区别是改变的速度和范围，而改变永远在发生。

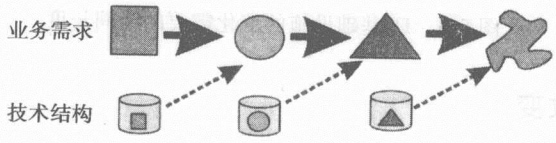


图5-1 业务需求持续变化，但是技术基础结构是很明确的

5.1 技术基础设施

技术基础设施位于业务之下，支撑业务的完成。因此当业务需求改变时，往往会出现问题。这是因为对技术基础设施的改变需要很大的努力，所以业务总是先于用来支持它的基础技术出现。

保持固定的技术基础设施有很多充分的理由。其核心是流行于系统厂商间的一个概念：当一项技术被定义之后，这个定义就是固定的了。这个基本的观念在很多情况下都会出现：

- 对于DBMS厂商，在一个项目开始定义数据结构的时候。
- 对于编译者，他们认为，一旦指定了处理过程和算法，他们将会按照这样的方法长期做下去。

- 对于商业智能厂商，他们认为一旦一个查询被执行，那么以后相同的查询就会以相同的方式执行。
- 对于管理者，他们认为，当他们作租约的或长期的承诺时，问题会得以解决并且不会演变成其他问题。

还有很多例子会错误地假设一旦需求确定下来就不会再出现其他的需求。

图 5-2 说明了这种设想。

但是需求在不断地改变，考虑图 5-3 的简单图表。

图 5-3 中菱形的阴影区域表示业务需求的改变。矩形区域表示 IT 部门改变技术基础设施以适应业务需求的改变。黑色虚线表示从业务需求改变开始到 IT 部门完成必要改变的时间长度。这条黑色虚线一般来说是很长的时间。图 5-3 所描绘的情况是普遍存在的。

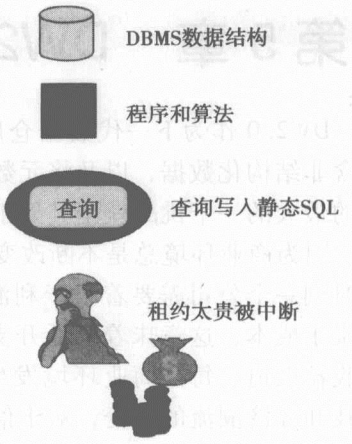


图 5-2 技术基础设施难以改变的众多原因

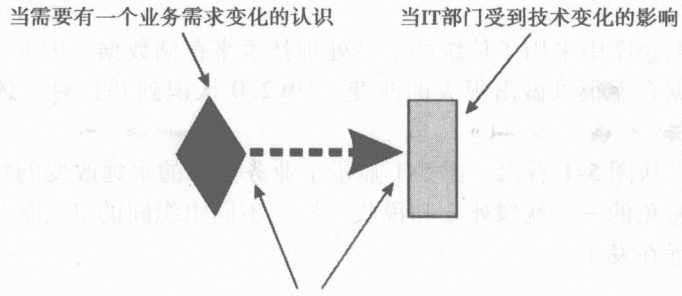


图 5-3 IT 基础设施的变化需要的时间长度

5.2 快速的业务改变

接下来考虑当业务改变速度比 IT 部门的响应速度快时会发生什么。图 5-4 描述了这种情况。

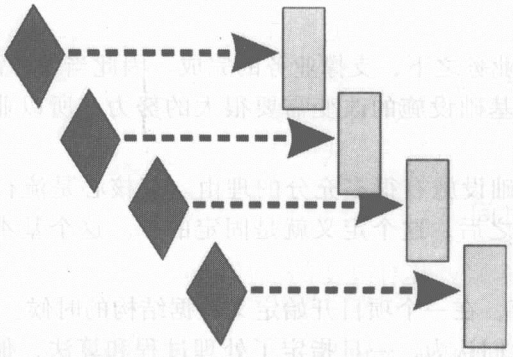


图 5-4 所需的改变速度高于可以承受这些改变的能力时会发生什么

图 5-4 描述了业务需求改变速度是如何高于 IT 部门对改变的响应速度的。当发现第一个改变时，IT 部门开始设计、计划并且建立响应。但是在他们完成之前，另一批业务需求开始出现，这些新的业务需求有它自己的生命周期。不同的另一组人员开始处理这个新需求。当两组人员必须处理和改变相同的数据和过程时，事情变得麻烦起来。而在第一个和第二个 IT 部门对应结构改变完成之前，第三批新的业务需求的出现使得事情变得更加糟糕。当第一、第二和第三组人员需要同时处理相同的数据和相同的过程时事情会变得真正复杂起来。

麻烦接踵而至。

5.3 环状改变

企业经常发现自己身陷一个恶性循环中。业务改变快于 IT 部门对改变的响应从而产生了新的改变了的业务需求，而这又产生一个永无止境的循环。图 5-5 描述了这个循环。

图 5-5 中的循环所带来的长远影响是 IT 部门被认为并没有对公司业务做出及时反应，业务和 IT 技术好像在两个方向上前进。图 5-6 显示了这一明显的分歧。

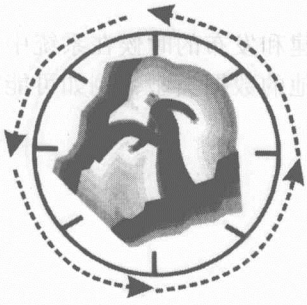


图 5-5 IT 部门陷入一个永远止境的循环

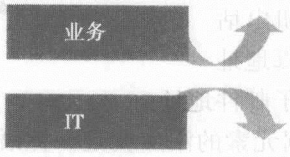


图 5-6 IT 部门和业务部门的分歧

5.4 打破循环

对于这种困境我们可以做什么？这里有几个可行的方案：

- 冻结业务需求：不幸的是，冰冻业务需求等同于一出现问题就把头埋入沙堆，不敢面对现实。
- 增加 IT 资源：在混乱中投入更多的 IT 人员是昂贵的而且通常是无效的（详见 Fred Brooks 的《人月神话》）。
- 缩短 IT 响应时间：缩短 IT 对新改变的业务需求的响应时间通常是唯一的选择。

事实上，就长远来说只有第三个选项是可行的。

5.5 缩短 IT 响应时间

图 5-7 表示 IT 需要缩短其对改变的响应时间是唯一现实的选择。

IT 对改变的响应时间必须缩短是一件事，而确定如何去做则是另外一件事。

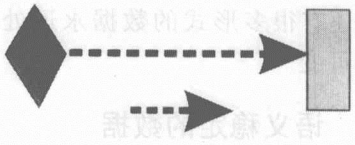


图 5-7 唯一现实的计划是缩短 IT 所需的对业务变化的响应时间

5.6 语义暂态、语义常态数据

缩短IT对技术基础设施的适应时间，以使其能适应不断进行的业务改变的最好最有效的方法之一在于一个让人意想不到的地方——语义暂态数据和语义常态数据。图5-8描绘了这两种类型的数据。

上面的矩形区域表示语义暂态数据，下面的区域便是语义常态数据。

常态语义和暂态语义是什么意思？数据类型可以在二者之间改变，数据内容可以改变。例如，我的银行账户可能在5 000美元到7 500美元之间保持平衡。这是一个数据内容改变的例子。但是，还有另一种基本的改变类型——语义数据改变。语义改变发生在数据定义改变时而不是数据内容改变时。举一个简单的语义改变的例子，假设已创建了一个账户持有者的数据定义。定义包括诸如：

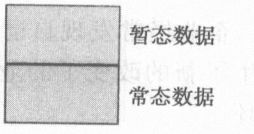


图5-8 在传统数据库设计中，暂态数据和常态数据是自由组合的

- 账户ID
- 开户人姓名
- 开户人地址
- 开户人生日

这个数据是在该例中的银行应用最开始被设计、构建和发布的时候在系统中定义的。

随着时代发展，人们意识到开户人数据还应包括其他的数据类型，例如可能需要为开户人定义添加以下类型的数据：

- 手机电话
- 传真地址
- 电子邮件地址

新数据元素的添加就是一种语义改变。

数据可以发生内容上的改变，也可以发生语义上的改变。本章剩余部分将致力于讨论语义改变而非内容改变。

5.7 语义暂态数据

语义暂态数据指的是那些可能要发生语义上改变的数据。有些形式的语义数据因其频繁的语义改变而臭名昭著。图5-9显示了一些语义不稳定的数据类型。

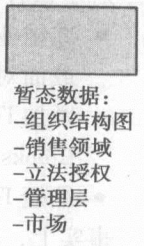


图5-9 暂态数据

组织结构图改变得出奇频繁。而每个新主管都认为重组公司是其职责。销售领域永远充满着重新洗牌。销售主管们总在争论俄亥俄州更适合哪里——东部地区还是中西部地区。一个主管则希望俄亥俄州在某个地方，同时其他主管则希望在另一个地方。

还有很多形式的数据永远处于语义改变中。数据是语义临时的，随处可变。

5.8 语义稳定的数据

与语义改变数据相反的是语义稳定的数据。语义稳定数据是常态数据——语义可能保

持长时间的稳定的数据。基本的销售数据是语义稳定数据的一个好例子。

图 5-10 描述了一些语义上稳定的数据。

基本的销售数据通常包括如下信息：

- 销售日期
- 销售总量
- 销售项目
- 向谁出售

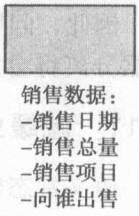


图 5-10 常态数据

这种基本销售数据在今天无疑是适用的。可以这样设想，4000 年前古罗马市场上的商人和中国北京的商人对同样的数据感兴趣，今天的沃尔玛也一样。

事实是，这一基本数据是必要的，并且在电脑出现前很久人们就对它感兴趣。并且可以预见，人们对这一基本数据在 2100 年仍然感兴趣，就像今天一样。

所有这一切都给出了语义稳定的数据存在的结论，在这里被称为常态数据。

那么，系统设计者和数据库设计者如何处理语义常态数据和语义暂态数据？他们根本就不关心这些。在数据库设计中，数据的语义并不是一个主要考虑的因素。这样做的一个直接结果就是，在数据库设计中语义常态数据和语义暂态数据通常是自由混合的。

5.9 混合语义稳定和不稳定数据

图 5-11 显示了自由混合语义常态数据和语义暂态数据的结果。

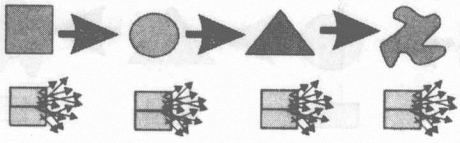


图 5-11 每次业务需求变化时，技术基础结构变得很混乱

图 5-11 的首行符号表示随着时间而不断改变的业务需求。每次业务需求发生改变，支持它的技术基础设施必须随之改变。语义常态数据和语义暂态数据是用于支持的技术基础设施中的常见组成部分，因此也必须适应不断改变的业务需求。所以，把语义常态和语义暂态数据混合在一起简直是自找麻烦。

图 5-12 显示了当语义常态和语义暂态数据混合后，就难以适应改变了。

无论何时，混杂在一起的数据如果发生改变将都会是大的变动，有不少充分的理由来解释这种情况。最重要的理由是任何变动都需经过一个数据转换的过程。考虑当改变发生时语义常态数据会发生什么。语义常态数据必须转变和调整，即使没有任何实际内容上的改变发生。而企业往往有大量语义稳定的数据的事实使得这种情况更加糟糕。并且当语义常态和语义暂态数据混合在一起时，还有许多其他理由证明改变会对混合在一起的这两种数据造成严重破坏。

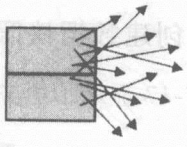


图 5-12 暂态和常态数据硬混在一起

5.10 分离语义稳定和不稳定数据

因此，问题自然就出现了。如果语义常态数据和语义暂态数据分离将会发生什么？图 5-13 描述了这种设计方法。

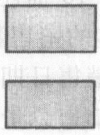


图 5-13 如果暂态和常态数据分离会发生什么

5.11 减缓业务的改变

语义常态数据和语义暂态数据的分离，缓解了通常情况下不断改变的业务需求所带来的破坏，如图 5-14 所示。

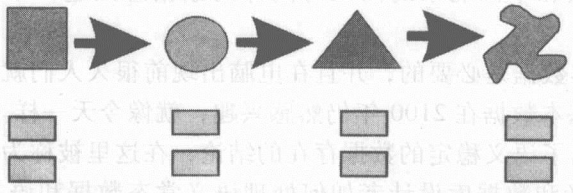


图 5-14 当暂态和常态数据分离，由变化造成的摩擦和混乱极大地缓解

尽管图 5-14 表明的现象是真实的，但为什么会这样却并不直观。有几个很好的理由可以说明分离语义常态数据和语义暂态数据对将 IT 技术基础设施从不断改变的业务需求中分离出来非常有利。考虑一下业务需求和语义常态数据一起改变。图 5-15 显示，语义常态数据受到业务需求的改变影响不大或者根本不受影响。语义稳定的数据在任何业务需求下在定义和性质上都是语义稳定的。

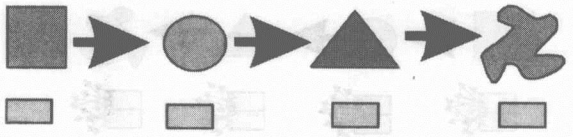


图 5-15 常态数据在变化中是稳定的

现在考虑当改变发生时语义暂态数据会发生什么。当语义暂态数据需要改变时，其实根本没有发生什么改变，而是创建了一个新的语义快照。创建一个新的语义快照比打开一个数据库来转换或改变它所包含的数据要容易得多。因此，当业务改变时，只是产生了语义暂态数据的一个快照（图 5-16）。

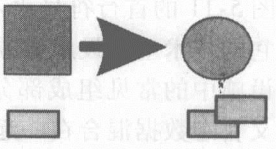


图 5-16 当变化发生时，一个新的快照被创建

5.12 创建数据快照

图 5-17 显示了业务需求随着时间的改变时，语义暂态数据会发生什么变化。

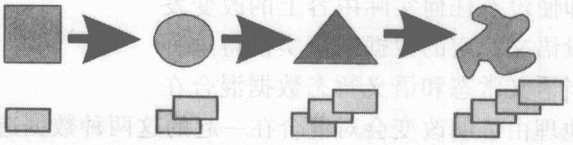


图 5-17 随着时间的推移，已收集的快照反映了随时间推移的全部变化

随着时间的推移，产生了一系列快照。每个快照是按时间分隔的，即每个快照有一个起始日期和终止日期。为了确定哪些是恰当的语义定义，查询必须有时间限定，这对任何一个查询都是很自然的。

图 5-17 表明通过采用对语义暂态数据产生新的快照而不是试图转换旧数据，管理那些改变就变成了一件非常容易做的事情。

5.13 历史记录

这种管理语义暂态数据改变的方法有一个附带的好处，就是创建了语义暂态数据的历史记录。记录如图 5-18 所示。

下面的例子突出了语义数据历史记录的价值。考虑一个对研究公司组织结构图的改变感兴趣的分析师所需的信息。假设分析师很希望看到该公司的组织结构图在 1990 年的情况。有了语义暂态数据改变的历史记录，分析师可以通过含有公司语义暂态数据的每个快照的起止日期很容易地确定和检索出该公司在 1990 年的组织结构图。

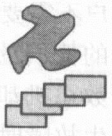


图 5-18 保持一段时间的数据快照的好处之一是有历史记录

当语义常态数据和语义暂态数据分离，并且这些数据形式成为技术基础设施的基础时，企业能够轻松地抵挡随时间而来的数据改变。这样，就减轻了由业务改变引起的系统动荡，如图 5-19 所示。

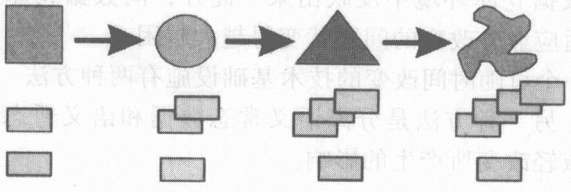


图 5-19 面对变化的业务需求从常态数据中分离暂态数据的结果

5.14 数据划分

下一个合乎逻辑的问题是如何划分数据。答案是语义常态和语义暂态数据在今后所有的数据库设计中应该被物理地分离。如果不行，还有一些技术来管理上述的 DW2.0 基础设施。

图 5-20 表示了基础设施管理软件如何管理 DW2.0 数据基础设施整体。

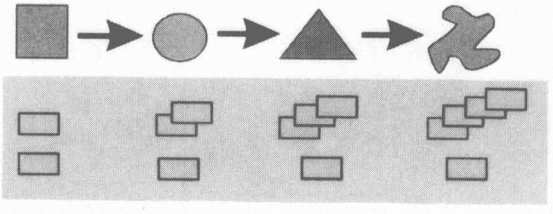


图 5-20 技术是一种管理常态和暂态数据的方式，如 Kalido

5.15 终端用户的观点

商业客户不是生活在技术世界里，而是生活在业务世界里。然而你能说得出的任何关于业务的事就是业务总是在改变。这种改变时快时慢，但改变是商业人士的一种生活方式。

经济的改变，法律的改变，新产品的出现和消失，竞争的改变，如此等等。

商业人士必须能够掌握信息来适应这些改变。如果信息基础设施不能适应这改变，那么它就会变成商业用户肩上的重担，信息则成为一种负担而非优势。只有当信息真正敏捷时，它才能成为企业的优势。

终端用户不需要知道“内部”是怎么回事。终端用户看到的信息基础设施就像大多数司机看到的他们的汽车一样。大多数司机知道有一个引擎，知道需要天然气和石油，但是绝大多数司机却不知道引擎的内部运转情况。对大多数司机而言，当汽车发动机盖下的引擎发生故障时，可以开向修车厂或者修理站。

这同样适用于商业分析师和 DW2.0。商业分析师意识到 DW2.0 的存在，但他不知道详细的支撑它的基础设施。所有的商业分析师都知道，当某些事情出错时，就是寻找一个了解 DW2.0 基础设施的数据构架师的时候了。

5.16 总结

DW2.0 的技术基础设施需要能够改变。当技术基础设施不可改变时，不久以后企业的业务需求就无法在数据仓库环境中反映出来。此外，向数据仓库添加的新需求所需的时间越长，数据仓库适应业务改变的问题就变得越大越困难。

为数据仓库创建一个可随时间改变的技术基础设施有两种方法。一个方法是使用专为这一目的设计的技术，另一种方法是分离语义常态数据和语义暂态数据。通过分离不同语义类型的数据，可减轻改变所产生的影响。

第6章　DW2.0的方法与途径

为了成功实施DW2.0，企业需要采用一种螺旋型的开发方法，以快速多次迭代的方式完成数据仓库的开发。每次迭代的开发周期不应超过3个月。这种螺旋式开发方法是人们在建立数据仓库时的一种标准惯例且已在世界范围内证明了它的价值。

那么螺旋式方法论到底是什么呢？为了解释这一方法，我们有必要先了解它的对立面，即瀑布式开发方法。图6-1展示了一个典型的瀑布式开发方法的生命周期。

瀑布式开发方法也被称为“SDLC”，是“systems development life cycle（系统开发生命周期）”的缩写。历史上，瀑布式方法在传递联机业务处理系统方面很成功。这些业务系统通常有很长的项目周期。它们是很大的项目，并且所有需求都是典型的预先记载。在数据仓库开发的早期阶段，即DW 1.0的阶段，人们错误地认为瀑布式方法能够用于数据仓库项目的开发。

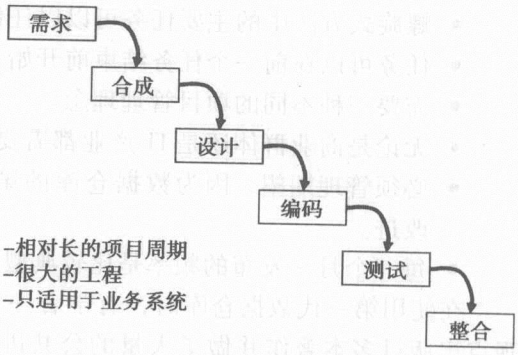


图6-1　瀑布式开发方法

6.1　螺旋式方法——主要特点综述

如图6-2所示，与瀑布式方法相比，螺旋式方法非常适合于那些不知道自己想要什么的用户。多数情况下，当建立一个数据仓库时是无法预先得到所有业务需求的。这不是某一个人的过错——不是因为商业人士拿不定主意，也并非因为信息技术工作组与客户过于脱节而无法了解其需求。这种现象很正常。商业智能的能力通常以一种不断发现的方式发展，商业群体的成员都会在最终看见时才能知道自己需要什么，而且一旦他们得到了自己需要的，他们就会理解这些并会有更多的需求。

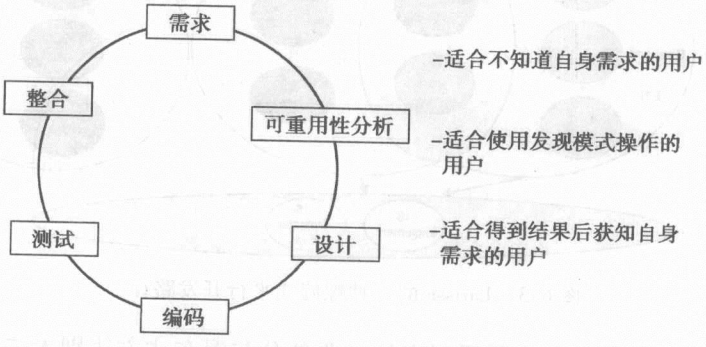


图6-2　螺旋式开发方法

商务智能需求的目标设置是不断变化的。这是可以理解的，因为业务也是在不断变化的。在一个业务系统的开发过程中，如果在系统发布后对一个业务请求进行修改，这将被视为失败的标志。相比之下，修改在数据仓库/商务智能系统中则被视为一件好事，修改需求被视为成功的标志，因为这意味着业务正在使用数据仓库，数据仓库激发了思考，并产生了对更多不同信息的需求。如果没有任何修改需求出现，数据仓库的主动性将是一个败笔。简言之，改变在交易处理系统中是不好的，而在数据仓库环境下是好的。

很多人将根植于面向对象技术的迭代方法与螺旋式方法搞混。虽然两者有很多相似之处，但它们是完全不同的两种方法。

以下是螺旋式方法的一些特点：

- 螺旋式方法扩展了原型的用途。
- 螺旋式方法中的主要任务可以以任何顺序出现。
- 任务可以在前一个任务结束前开始。
- 需要一种不同的项目管理理念。
- 无论是商业群体还是IT产业都需要做出文化上的改进。
- 必须管理期望，因为数据仓库的第一次迭代开发不是很完整，还需要进一步的改进。
- 每三个月一发布的频率是比较典型的，需要严格的范围控制。

在使用第一代数据仓库时，有学者一直呼吁使用螺旋式方法。Larissa Moss曾关于该观点出版过多本著作并做了大量的公共讲座。有的团体因她的指导受益，而忽视该方法的团体仍处于煎熬之中。当DW2.0的时代到来时，从过去的错误中吸取教训是很有必要的。

图6-3阐述了Larissa Moss的“商务智能路线图”方法。

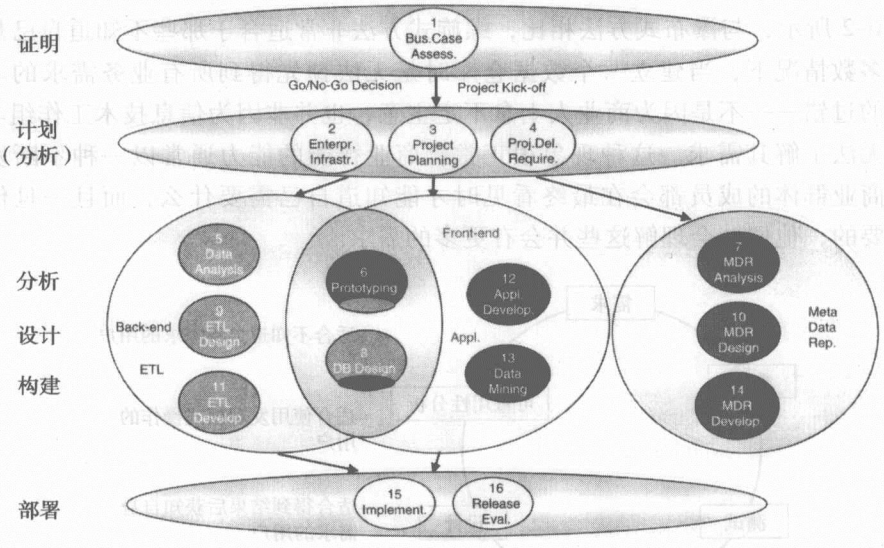


图6-3 Larissa的三种螺旋式平行开发路径

乍看起来，在图6-3中左边所罗列出的主要部分与瀑布式方法别无二致。事实上，人们在瀑布式项目中做的很多工作也同样适用于螺旋式方法。然而正如图中所强调的，在

分析、设计和构建等作业部分之间有很高的并发性。将其与以下事实联系起来，即作业可以在螺旋开发生命周期的任意一点开始（例如，先从构建开始然后向上继续是很常见的），用户可以看到这种方法与传统的IT开发方法有着极大的不同。

需要对团队组织进行深思熟虑以完成并发任务。图6-3突出了三种自然的作业分组：后端作业、前端作业以及元数据作业。该图描绘了前端/后端作业关于原型和数据库设计的重叠部分。元数据作业虽然出现在图例的右边，但事实上与其他的平行作业有高度的相互依存关系。元数据作业添加量超过前端作业和后端作业会更加精准，但同时也降低了可读性。项目管理必须能够识别并管理这种相互依存的关系。

值得注意的是，尽管一个数据仓库应用的迭代可以从构建开始，但是除非直到证明、计划、分析、设计、构建和部署等步骤全部完成，否则这种迭代是不完整的。要想建立一个应用，通过这三种螺旋式作业路径可能会需要几个步骤。虽然理论上应用的每一个步骤都要完成，但通常不必重新访问“证明”，甚至也不必完成每条路径上的一些其他步骤。

图6-4可以让我们对三种平行螺旋式开发过程有更深刻的理解。

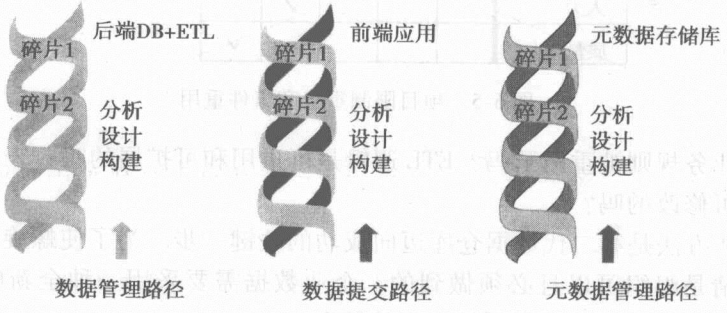


图6-4 Larissa的三种平行螺旋式方法开发路径

螺旋式方法利用几个临时的可交付成果，或者称为“碎片”，来产生一个应用。值得注意的是，在图6-4中，每一个螺旋碎片都在整个应用范围内扮演细小但意义非凡的角色。每个碎片都无法一次性完成构建。图中展示了每种作业路径的两个碎片，事实上每种路径都有多个碎片且数量几乎不会相同。

每个碎片的范围必须保持尽可能小，另外要非常小心地管理范围，以免出现范围大到不可控制的变化。

螺旋式方法的目标是建立可重用资源的详细清单。Larissa在图6-5中阐述了如何动态地重新规划项目限制，以实现质量最大化和范围最小化。

使用螺旋式方法会帮助团体从废件和返工的死循环中解脱出来，从而向着资源重用的文化风格迈进。这种新的思想着眼于可重用资源满足业务需求，以便更快更好更廉价地完成任务。

第二代数据仓库需要抛弃传统的投资回报率的方式（ROI）。这种方式导致了许多组织建立相继的“单点解决方案”（或者数据集市），很少或根本不关注任何已提交成果的可重用性。ROI的宗旨是：“越快完成越好，并取得一定回报。”实际上最初的成功是不能持久的，并且数据结构相当脆弱（不可改变），这一点似乎不符合ROI思维模式。

在DW2.0中，成功取决于总资产收益率（ROA）。数据被重用了吗？元数据被重用

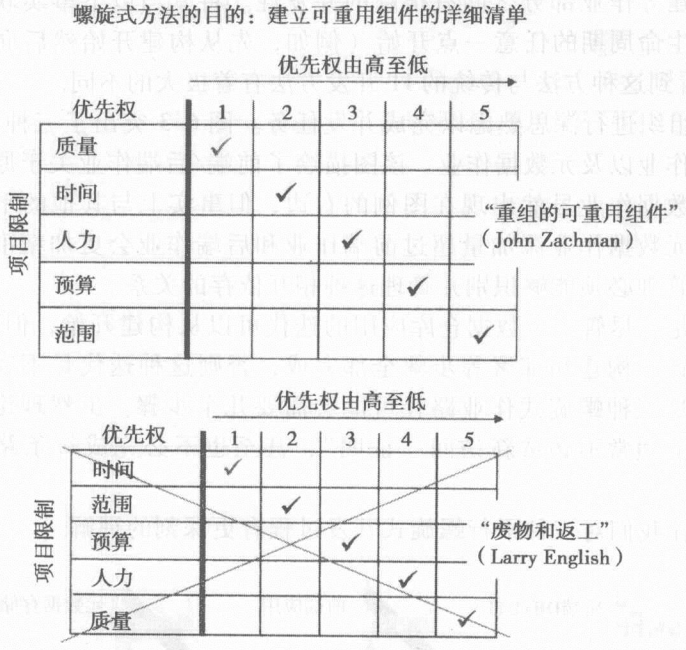


图 6-5 项目限制重组和组件重用

了吗？典型的业务规则被重用了吗？ETL 逻辑是可重用和可扩展的吗？表示层的所有组件都是可重用和可修改的吗？

引入螺旋式方法是第二代数据仓库迈向成功的关键一步。为了使螺旋开发方法获得成功，有一些事情是组织可以且必须做到的。企业数据需要采用一种全新的方法。本章的剩余部分将讨论这一方法，即所谓的“七流法”。

6.2 七流法——总览

一个基本的前提是商业智能策略应被看作是一个程序，而并非项目。这样的商业智能策略应该随着业务需求变化而不断地得到磨砺，变得越来越复杂。要在不断变化的业务环境中实现和保有这一策略，要求提供更多的解决方法，不仅仅是“盒子里的数据仓库”或是 30 天到 90 天的“神奇的解决办法”。

那么，为了在商业智能/数据仓库程序中实现持续的成功，到底需要哪些因素呢？下文将给出一个高度总结的答案。从图 6-6 开始解释了七流法——一个已经被证明了的商业智能计划和发布框架。

理解“七流法”的关键是这样一个事实：每一个事件流是根据不同的步调行进的。每一个流是同步开始和并发驱动的，并且需要监控和协调。图例的组织是无先后顺序的。

6.3 企业参考模型流

第一个事件流建立了一个企业数据模型（图 6-7）并持续地对其进行维护。当然这并不是通过说“让这个世界停止吧，我要建立一个巨大的企业数据模型”来实现的，而是按对象域增量建立的（例如，消费者、产品等）。

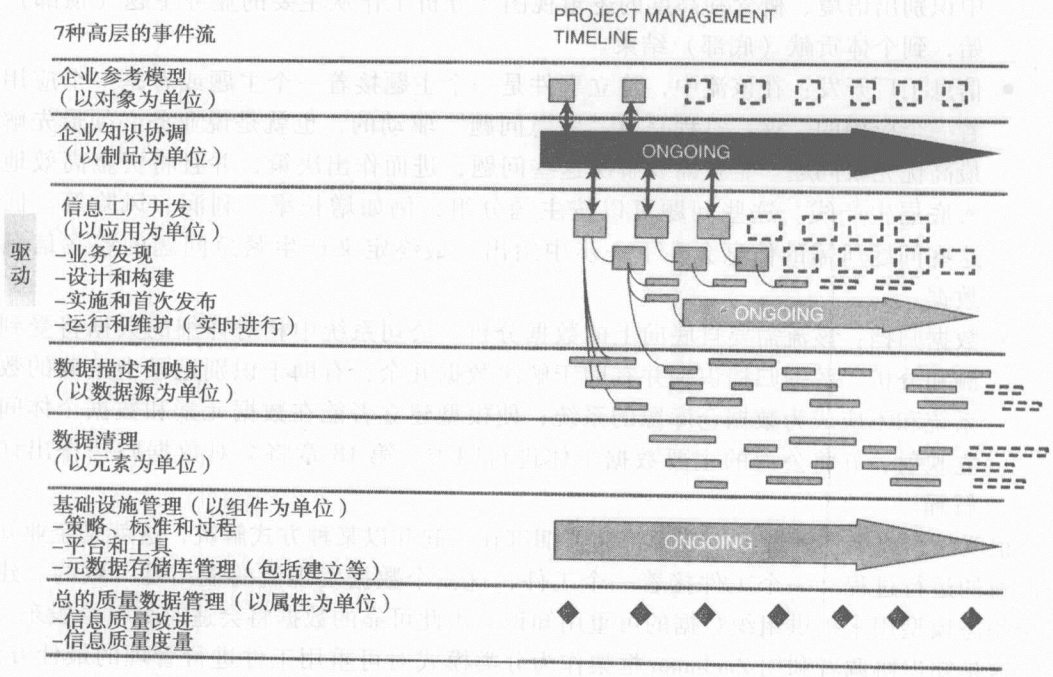


图 6-6 DW/BI 项目中的“七流法”

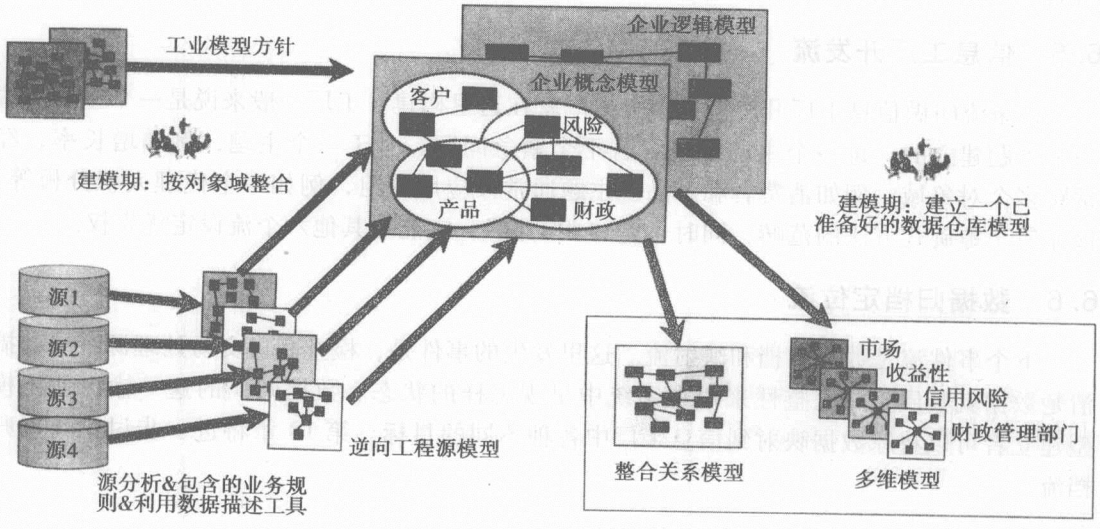


图 6-7 企业数据模型流

6.4 企业知识协调流

下一条事件流，即知识协调流，承担着从三个数据发现流（例如，企业数据模型，信息工厂开发，数据归档）中获取不同的产品并理解这些发现，如下所示：

- 企业数据模型：企业数据模型通常需要自顶向下地分析企业数据——从企业数据

中识别出语境、概念和高度的逻辑视图。分析工作从主要的业务主题（顶部）开始，到个体贡献（底部）结束。

- 信息工厂开发：在该流中，建立事件是一个主题接着一个主题或者是一个应用接着一个应用的。这一过程是由“紧急问题”驱动的，也就是说业务必须最先解决最高优先权问题。企业需要解决这些问题，进而作出决策，并且将资金高效地投入底层生产线。这些问题可以按主题分组，例如增长率、利润、风险等。回答这些问题所需的信息会在下一步中给出，最终定义产生紧急问题的答案信息的数据。
- 数据归档：该流需要自底向上的数据分析。公司系统中相对详细的数据将受到检测和分析。数据归档识别并有助于解决数据冗余；有助于识别记录中正确的数据系统和不应成为数据仓库源的系统；使模型建立者能在数据元素和数据个体间建立映射，并将公司的主要数据个体进行归类。第18章将会对数据归档作出详细解释。

很明显，以上三种数据发现源都需要捆绑在一起并以某种方式解决，这就是企业知识协调流的运行过程。一个工件接着一个工件，这三个数据发现流的输出是一致的。建立一个稳态模型用来提供组织数据的可重用知识，由此可靠的数据将会适时交付给股东。

这是知识协调者利用Zachman框架作为分类模式对可重用工件进行管理的最佳方法。图6-8阐述了利用Zachman框架背景作为“思考工具”时，知识是怎样自顶向下和自底向上进行协调的。

6.5 信息工厂开发流

这条流叫做信息工厂开发流。信息工厂就在这里构建。工厂一般来说是一个主题接着一个主题建立的。每一个主题都包含若干个紧急问题。对于一个主题，例如增长率，经常跨多个对象域，例如消费者和产品。主题通常按应用分组，例如一个代理人积分板等。该流属于螺旋式方法的范畴，同时也是“驱动流”，它能对其他六个流设定优先权。

6.6 数据归档定位流

下个事件流是数据归档和映射流。这里发生的事件是，检查联机交易处理源系统，搞清楚数据就其质量与完整性来说在系统中呈现怎样的状态。数据归档的这一输出使得模型建立者可以将原数据映射到信息工厂中各种不同的目标。第18章将进一步讨论数据归档流。

6.7 数据纠正流（旧称数据清理流）

该事件流是按属性审查相应的源系统，决定哪些数据需要被纠正、补充或清除，并决定采用何种纠正法则。这一部分也将在第18章具体讨论。

6.8 基础设施流

如图6-9所示，该流主要关注对信息工厂可扩展性的架构支持，包括对于人力、资源、平台、工具、政策、标准和过程的考量。该流是一个组件接着一个组件开始的。

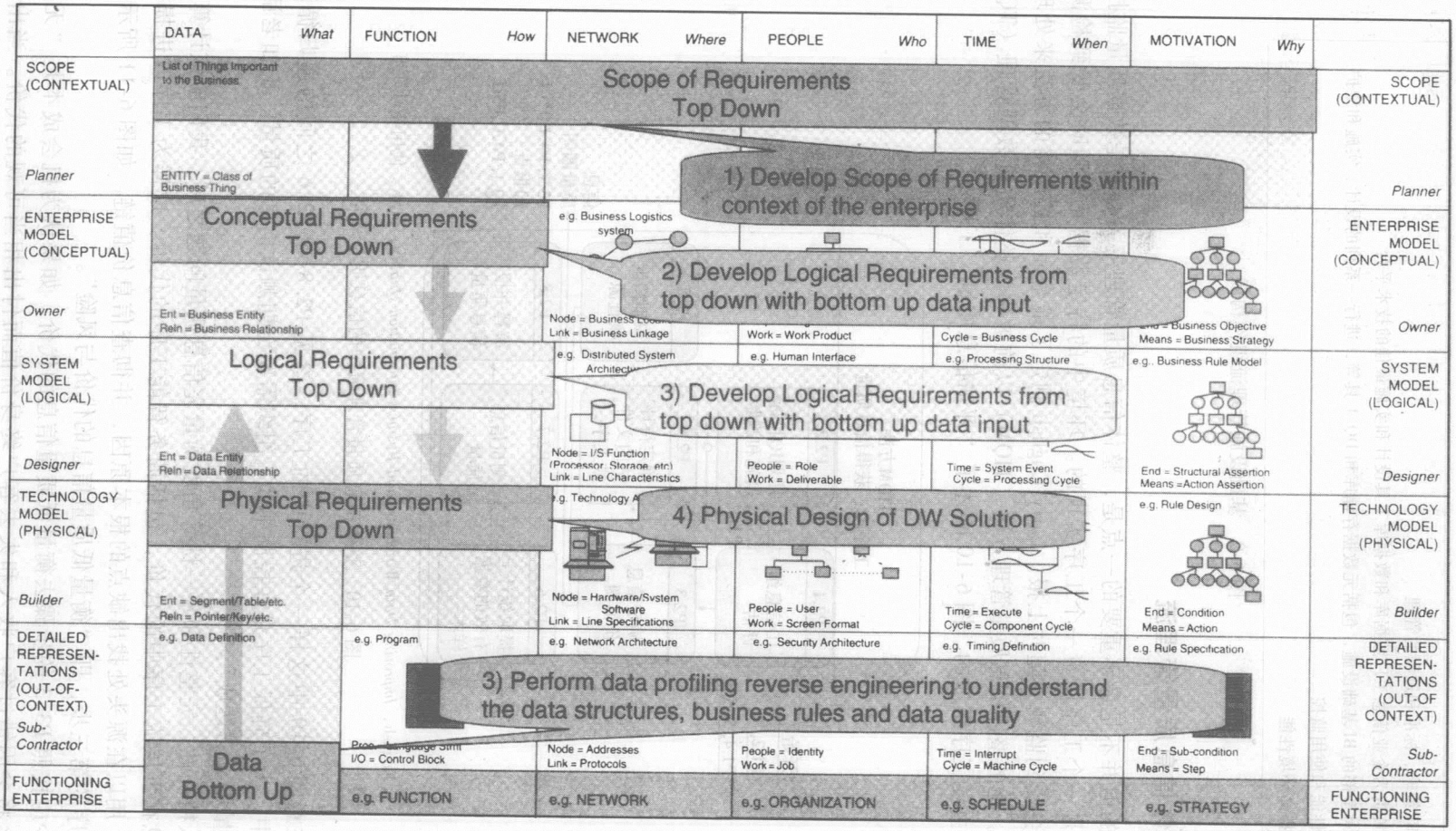


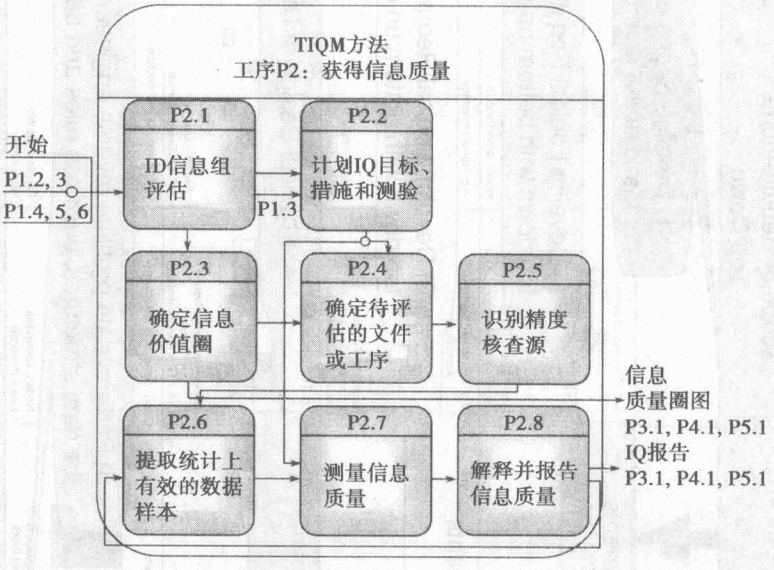
图6-8 知识协调流利用Zachman框架作为分类模式

- BI策略、标准和过程的管理
- 负责为企业信息工厂数据库和数据库工具设计和实施最佳的技术平台
- 对全部的BI基础设施，包括元数据存储库和DQ工具等，进行一系列的设计、实施和维护
- 性能和使用监控
- 提高环境性能

图6-9 基础设施管理流组件构成

6.9 整体信息质量管理流

最后的但并不是最不重要的一点是，整体信息质量管理流关心的是数据质量监控和工序改进，是一个工序接着一个工序实现的。环境中的特定数据元素随时会受到检测，并且其质量会受到监测并随时上报。Larry English制定了最全面和最严格的方法来处理信息质量管理。他的整体信息质量管理方法（TIQM）以前被称为整体质量数据管理（TQdM），是由几个主要工序组成的。图6-10阐述了“评估信息质量”工序。



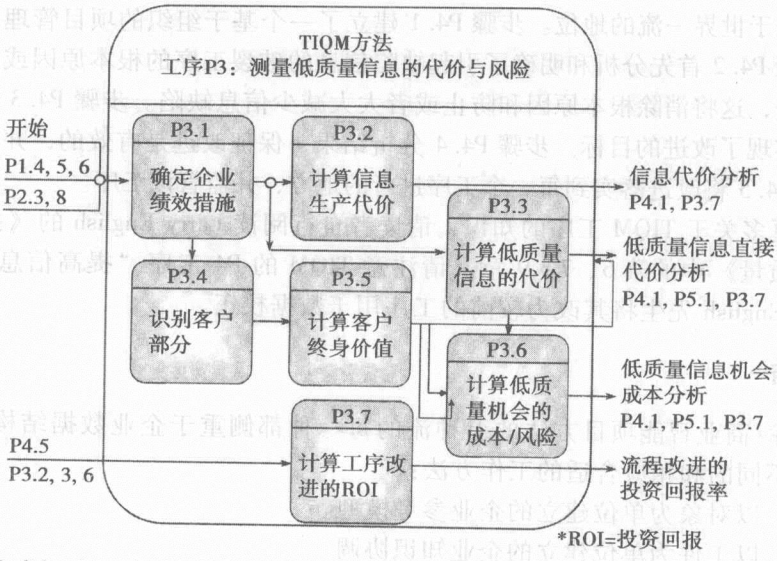
L. English, *Improving Data Warehouse and Business Information Quality*, p156, 使用经过许可

图6-10 整体信息质量管理流中的信息质量评估

请注意图6-10中这样一个进行中的工序，参见P2.8和P2.6之间的递归循环。在DW2.0中，为了确保工序是在控制中的，组织需要定期测试信息的质量。正如老话所说，“不做考量则无法达成”。

在从精确性、完整性和唯一性等角度衡量完信息质量问题之后，我们应该计算一下低质量信息在下层流工序中的代价，包括商务智能工序的代价。这样为工序改进提供了业务案例，用以在源头处找出缺点的根本原因，并彻查信息价值链。如图6-11所示，这是TIQM工序的第三步，即“衡量低质量信息的代价与风险”。

至少应按照P3.3的步骤去衡量低质量信息的代价。如果认为机会成本是“无形”的时，那么将会因错过客户收入和失去客户终身价值而付出相当可观的代价。当出现以下

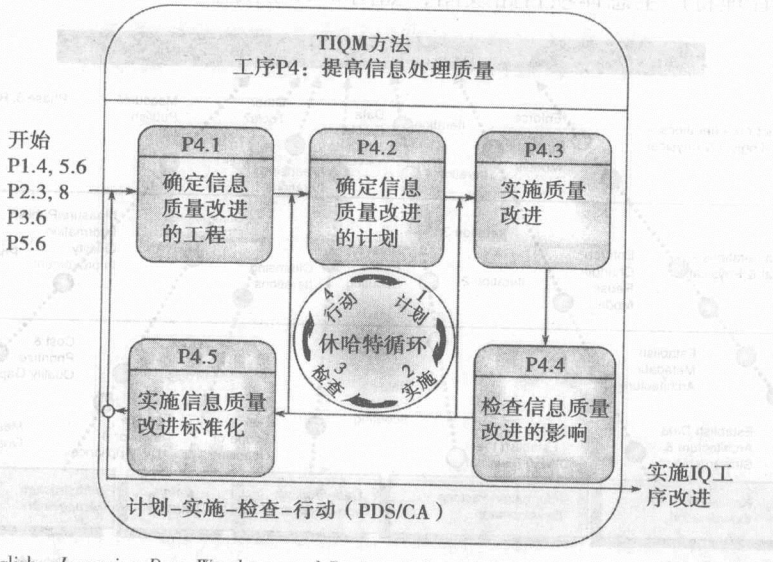


L. English, *Improving Data Warehouse and Business Information Quality*, p214, 使用经过许可

图 6-11 测算低质量信息的代价以及 TIQM 工序改进的投资回报率

低质量信息时，例如，姓名拼写错误、地址错误、客户记录丢失、记账错误以及物品投递错误等，都会引起实际的财产损失。

在搞清楚低质量信息的代价后，我们能够着眼于工序改进的高回报区域。通常采用的改进方法是 Pareto 方法——从最重要的到次重要的，依次类推。“计划 - 实施 - 检查 - 行动改进循环”这一工序在 TIQM P4 步骤中给出，“改进信息处理质量”在图 6-12 中给出。



L. English, *Improving Data Warehouse and Business Information Quality*, p290, 使用经过许可

图 6-12 TIQM 中改进信息处理质量工序

工序 4 是 TIQM 的核心工序，需要在“信息质量管理”标签中使用“质量”这一字眼。正是这一工序，消除了引起过程失败、信息碎片和重写的缺陷。当某组织习惯使用这一工序

后，它将会处于世界一流的地位。步骤 P4.1 建立了一个基于组织的项目管理准则的工序改进建议。步骤 P4.2 首先分析和明确了引起缺陷信息的破裂工序的根本原因或原因，然后定义了流程改进，这将消除根本原因和防止或者大大减少信息缺陷。步骤 P4.3 实施改进，并观察和保证实现了改进的目标。步骤 P4.4 分析结果，保证改进是有效的，并将已有的经验存档。步骤 P4.5 将改进落实到每一个工序进行的地方，并监管该工序。

欲了解更多关于 TIQM 工序的知识，请读者自行阅读 Larry English 的《提高数据仓库和商务信息质量》中的第 6、7、9 章。请注意 TIQM 的 P4 工序“提高信息处理的质量”编号为 P5。English 先生将其改为较前的工序用于数据校正。

6.10 总结

数据仓库/商业智能项目方法的七种流的每一种都侧重于企业数据结构的不同方面，并且都采用不同的和相应合适的工作方法：

- 流 1：以对象为单位建立的企业参考模型。
- 流 2：以工件为单位建立的企业知识协调。
- 流 3：以主题为单位建立的信息工厂开发。
- 流 4：以源为单位建立的数据归档和定位。
- 流 5：以属性为单位建立的数据校正。
- 流 6：以组件为单位建立的基础设施管理。
- 流 7：以工序为单位建立的整体信息质量管理。

每个流以不同的比率生产可交付物。精明的 DW/BI 项目管理者将认识到这些不同比率和节奏的存在，将每个并发流的工作优先权进行同步，并向组织发布有意义的信息。DW/BI 项目管理将产生总体项目路线图，如图 6-13 所示。

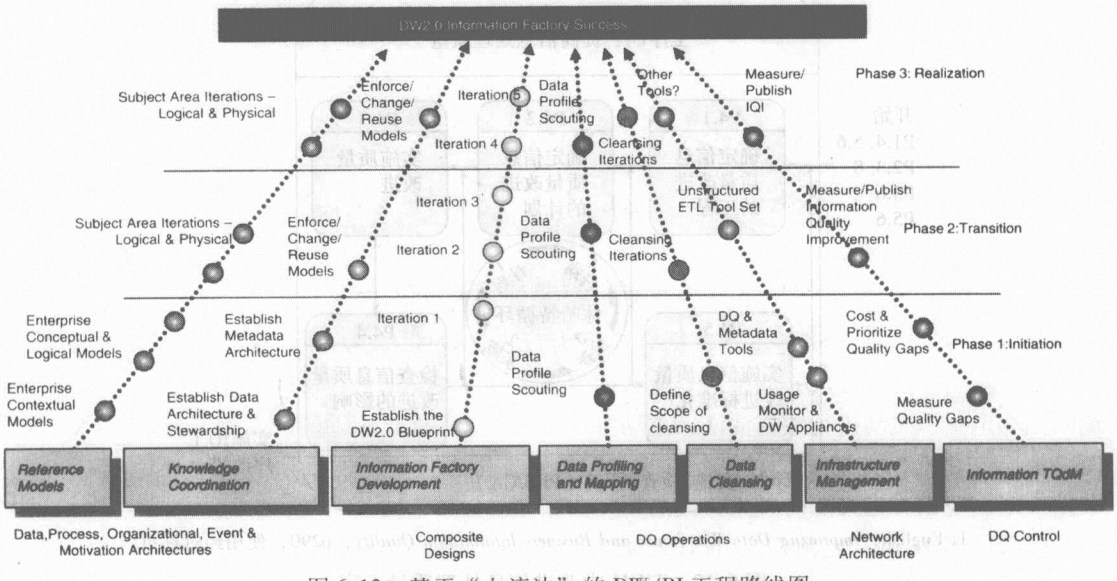


图 6-13 基于“七流法”的 DW/BI 工程路线图

“七流法”是设计 DW/BI 项目的一个框架和工具，并有助于自身适应快速的螺旋式开发。“七流法”和螺旋式开发方法的相互作用在以下几个图表中得到生动的描绘。

图 6-14 描述了螺旋式开发方法在信息工厂开发流中的位置。

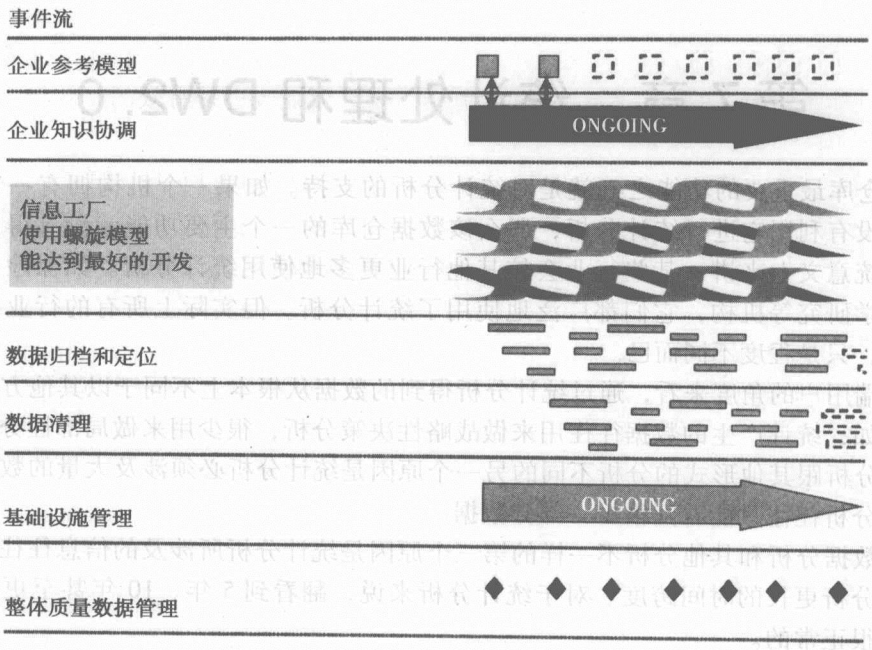


图 6-14 DW/BI“七流法”中的螺旋式开发方法。如何将螺旋式开发方法与“七流法”融合在一起

图 6-15 描述了“七流法”和螺旋式开发方法之间的关系。

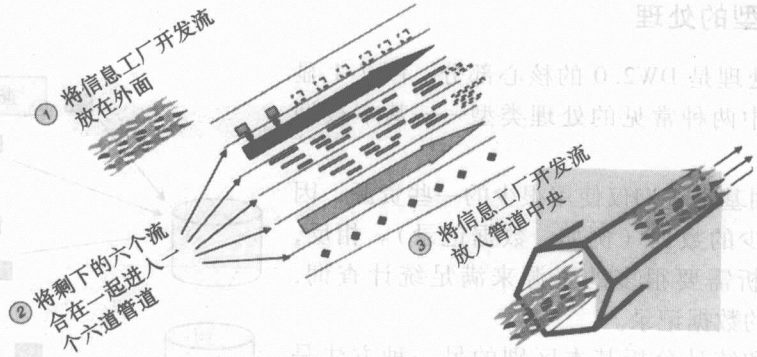


图 6-15 “七流法”——为螺旋式开发方法作准备

螺旋式方法已被证明是建立数据仓库最为行之有效的方法。螺旋式方法因为“七流法”的实施，其功能更为强大。如果对企业数据模型足够关注，那么，协调企业知识、主动进行数据归档和定位、主动进行数据清洗、主动管理基础设施、建立正确的全体数据质量管理文化则可以加快螺旋式 DW/BI 开发的迭代。这种组合方法能够使得开发团队不会触及众多路障——数据模型、规则和定义随时可用，提前得知数据质量异常，适当的基础设施已然到位。

螺旋式开发和“七流法”的好处是为季度发布的业务提供规律的数据。目前的挑战是无论是 IT 行业还是商务团体都必须在文化方面做出改变。做得最成功的组织是通过螺旋式方法与“七流法”相结合的培训，加之对两种方法皆有深刻经验者的指导，来实现这种文化变革的。这种组合的 DW/BI 项目方法也必须以正确的管理结构为支撑。

第7章　统计处理和DW2.0

数据仓库最重要的功能之一就是对统计分析的支持。如果一个机构拥有一个数据仓库，但却没有利用它进行统计分析，那么该数据仓库的一个主要功能就没有得以开发利用。从传统意义上来讲，某些行业会较其他行业更多地使用统计分析。如保险、手工制造以及医学研究等机构，它们都广泛地使用了统计分析。但实际上所有的行业都使用了统计分析，只是程度不同而已。

从终端用户的角度来看，通过统计分析得到的数据从根本上不同于以其他方式得到的信息。例如，统计产生的数据往往用来做战略性决策分析，很少用来做局部性分析。

统计分析跟其他形式的分析不同的另一个原因是统计分析必须涉及大量的数据，而其他形式的分析往往只能看到很少一部分数据。

统计数据分析和其他分析不一样的第三个原因是统计分析所涉及的信息往往具有较其他形式的分析更长的时间跨度。对于统计分析来说，翻看到5年、10年甚至更久之间的数据都是很正常的。

DW2.0支持统计分析和处理，就像它支持其他形式的分析一样。根据统计分析的类型和使用频率的不同，DW2.0可以直接或者间接使用。

7.1　两种类型的处理

统计分析处理是DW2.0的核心部分。图7-1显示了数据仓库中两种常见的处理类型——基本查询和统计分析。

图7-1表明基本查询仅使用很少的一些资源，因为其仅需要很少的数据（例如，数据记录）。相反，典型的统计分析需要很多的资源来满足统计查询，所以需要大量的数据记录。

看待查询和统计分析基本区别的另一种方法是比较一个典型的查询处理和一个典型的统计分析的输出，如图7-2所示。

图7-2表明一个查询仅查找并给出少量数据。在这个例子中，查询得到了Luiz Pizani的个人记录，通过这条记录我们找到了他的银行账户，为满足这次查询，需要分析的数据也仅仅是一条或两条数据。

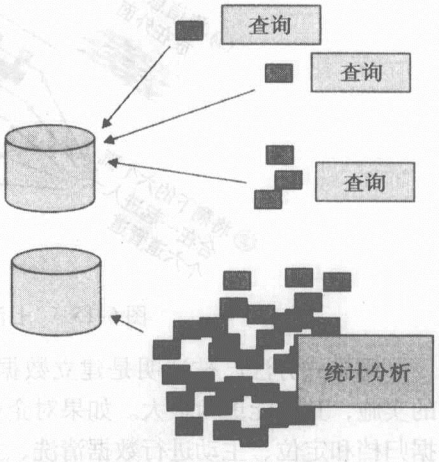


图7-1　统计分析中用到的资源远远多于其他形式的处理

查询 – Luiz Pizani，周二一天的记录为$2568.08

统计分析 – 通过25 998个测试实例所得的均值为22 190

图7-2　基本查询得到的结果和统计分析差别很大

而在统计分析中，毫无疑问需要大量的数据记录。如图7-2所示，查询的结果涉及统计平均值或者平均数的计算，要算出这个平均值，就要访问近26 000条记录。进一步讲，统计查询所需的全部记录需要一次性访问。直到所有记录可用时才开始计算平均值。

7.2 使用统计分析

利用统计分析可以做很多事情。最简单的统计分析可以是建立一个数据分析文件。数据的分析是指数据实体内容的统计总结。数据统计分析可以回答以下典型的问题：有多少条数据记录？最大值或最小值是多少？平均数、中值、最频值又都是多少？有没有超出指定范围的值？是否存在指定范围内的边缘值？这些数据值的分布有什么规律？

所有诸如以上的这些问题都可以添加到数据实体的分析中。数据实体的分析可以使分析员看到数据集的概述——就好比看到一片森林，而不是单独的树木。

数据的统计分析除了以上的用途之外，还有许多其他方面的应用。例如对企业数据和外部数据的比较。其中，企业数据的生成和发展是比较的第一步，然后再捕捉外部数据并将它们置于相同的环境条件中。然后就可以完成比较了。

为了解释典型的企业数据与外部数据的比较，图7-3给出了很好的例子，它描述了可口可乐公司的数据与整个饮料行业销售额数据的对比情况。

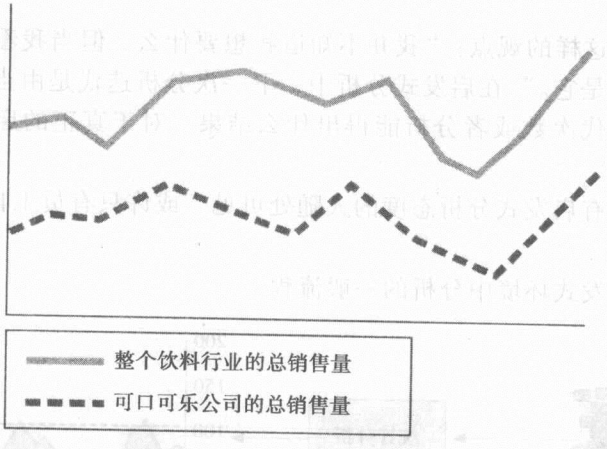


图7-3 企业信息与行业信息的比较

对于饮料行业分析员来说，最感兴趣的事情莫过于比较统计中的高峰与低谷。他们往往想知道在全行业呈现一种下降的趋势下可口可乐的销量是否还在增长？或者是在全行业呈现一种上升的趋势下可口可乐的销量是否却在下降？在这两条销售额曲线之间是否存在着一种通用的关联模式？

拥有对于企业信息和外部信息进行比较的能力可以使我们真正拥有宝贵的商业洞察力。

7.3 比较的完整性

数据的有效性是比较统计分析的一个关键问题。严谨的统计学工作者一般会确定他们比较的对象是否为同一事物——是苹果和苹果比较还是苹果和橘子比较。回到图7-3的例子，饮料行业的销售额与可口可乐公司的销售额相比较是否是合理有效的？如果饮料销

售包括啤酒和葡萄酒，那么将可口可乐和其比较是否还是公平且有意义的呢？如果可口可乐包括诸如美汁源的饮料呢？那比较美汁源和百事可乐的销售额是否是合理（或者明智）的呢？

外部数据和内部数据的比较引入了一些问题，这些问题都得在统计比较可以被认为是合理有效之前加以解决。

除了对内部和外部数据的比较之外，统计分析还有许多其他重要的用途。其中一项就是确定数据发展趋势和数据的模型。

支持统计分析的数据仓库中的业务案例就是一个非常好的例子。

分析无处不在，甚至在考虑统计分析本身时都需要分析活动。简单查询的目的就是找到能马上满足需要的信息。然而，当出现统计分析时，它一般会采用一种完全不同的分析形式，称之为启发式分析。

启发式分析是这样一种分析，它属于发现过程中的一部分。一个事物可以被认为是启发式的，如果它有助于我们学习、发现或者解决问题。在发现过程中，分析员并不知道数据中隐含的信息。他们要在并不知道数据包含什么也不知道自己期待什么的情况下开始去挖掘或者学习数据的内容。

7.4 启发式分析

启发式分析员有这样的观点："我并不知道我想要什么，但当我看见它的时候我就会知道是不是想要的就是它。"在启发式分析中，下一次分析迭代是由当前分析的结果决定的。计划好分析的迭代次数或者分析能得出什么结果，对于真正的启发式分析来说是不可能的。

商业领域中，抱有启发式分析态度的人随处可见。或许只有员工自己真正知道自己的所需。

图 7-4 描述了启发式环境中分析的一般流程。

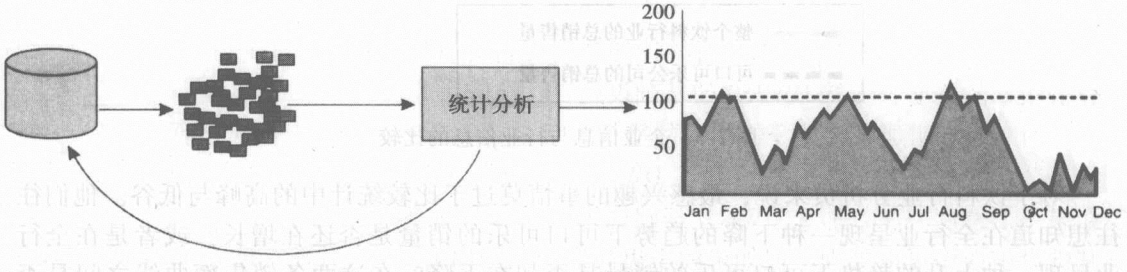


图 7-4 统计处理的一个特点是启发式分析

统计处理和启发式分析存在着一定的关联性。在启发式分析中有这样一个情况，但其不常见，就是偶尔地进行数据冻结。数据冻结后，系统便不能吸收新的数据。启发式统计处理需要偶尔这样做的原因是，我们需要检查分析的结果是由算法还是数据的改变产生的。

例如，一个分析员针对一个数据实体运行了一次分析，发现数据返回的平均值为 67。接着，他改变了所用的算法并再次运用该分析，这次返回一个新的平均值 98。这时的问题就是，分析结果的改变是由算法功能改变还是由数据的变化造成的？如果第二个分析

在一个不同的数据集下进行，那么分析结果的改变很有可能是数据的不同引起的，而不是由计算过程中所使用算法的改变引起的。

7.5 冻结的数据

当操作后的结果有了较大变化时，需要冻结计算中所使用的数据。冻结数据意味着结果的改变可以肯定地归结为算法的变化而非其他的原因。

图7-5描述了用于支持启发式处理和结果分析的冻结数据。

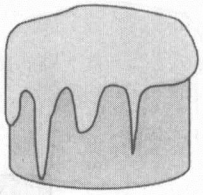


图7-5 有时候数据需要被冻结

7.6 探索型处理

统计处理的本质之一是它经常进行反复的探索过程。在许多类型的信息处理中，分析过程往往建立在内容、形式和结构都已知的数据上。而另外一些类型的信息处理则恰恰相反，它们往往对数据的内容、形式和结构一无所知。而统计分析就特别适合这用类型的分析——探索型处理。

图7-6描述了探索型处理的过程。

DW2.0数据仓库环境所面临的挑战之一就是如何最好地支持统计处理。当然DW2.0拥有支持统计处理所需的重要数据。事实上，DW2.0架构中包含了统计分析的关键组成部分。然而，关于如何使用DW2.0环境下的数据仍然有很多需要解决的问题。

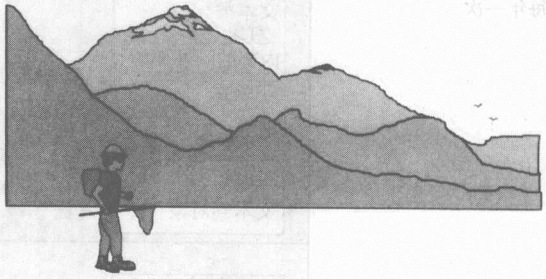


图7-6 探索过程——找出可能发生的事件

7.7 分析频率

统计分析的频率和DW2.0对统计处理的支持有着密切的关系。图7-7表明随着统计分析频率的改变，支持DW2.0的基础设施也会相应变化。

图7-7表明随着统计分析频率的增长，对单独的探索工具的需求也就越来越大。如果统计分析一年只做一次，那么基本DW2.0的基础设施就可以单独胜任这个水平的处理。如果一个季度一次，那么DW2.0基础设施可以勉强处理。如果一个月一次，那么DW2.0基础设施或许可以处理。但如果频率更高而DW2.0构架又没有额外的性能增加，那么便无法处理了。当然在很多机构中统计分析常常是一小时一次，那么需要将单独的探索工具添加到数据仓库中，以保证分析处理的正常进行。

7.8 探索工具

探索工具提供了这样一个平台，在它上面可以进行统计处理并对核心DW2.0基础设施不会产生影响。探索工具和DW2.0环境在物理空间上相分离，它们位于不同的物理平台上。

如果需要的话，探索工具可以被冻结一段时间。如有必要它还可以包含外部数据。典

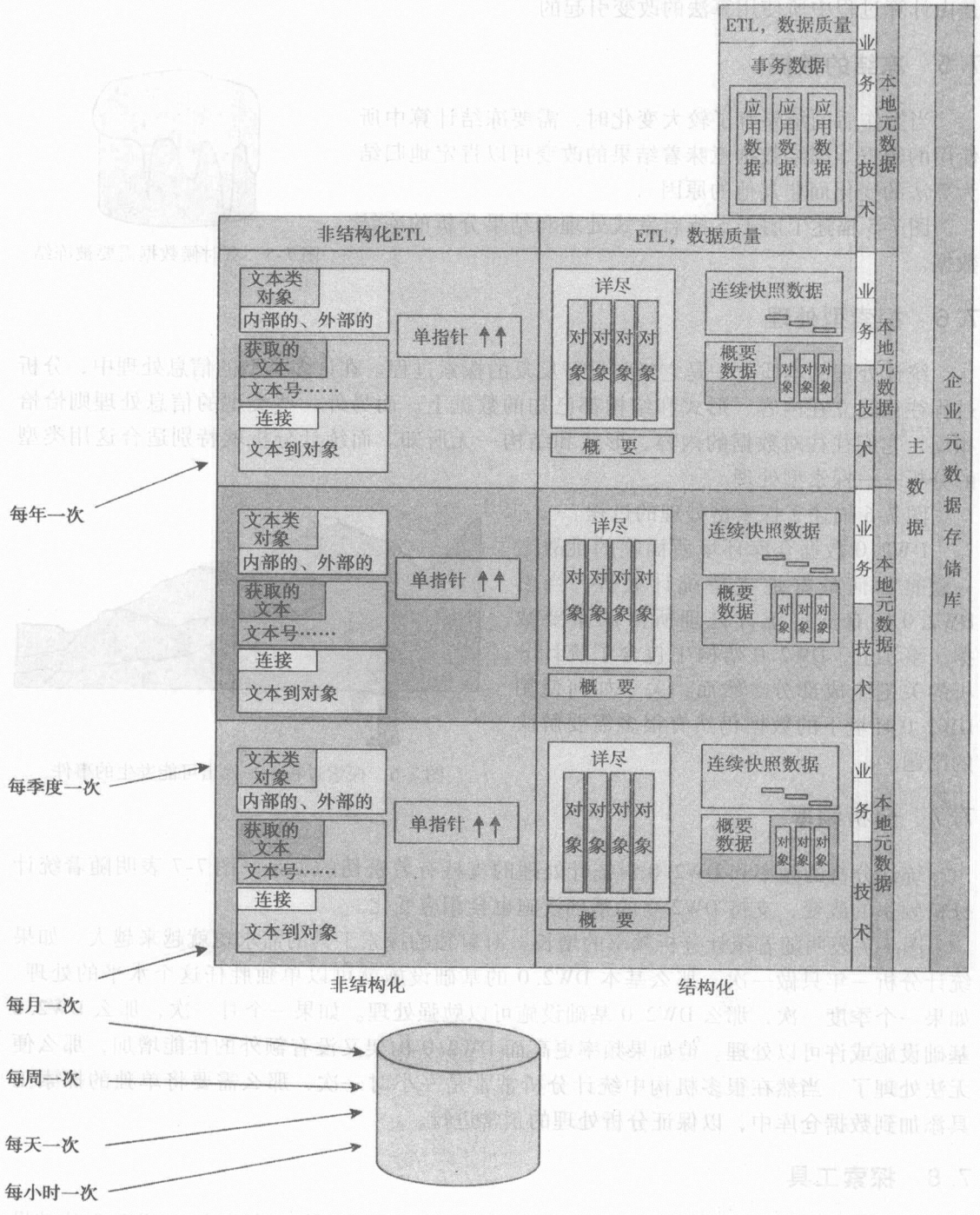


图 7-7 统计分析的频率决定是否需要单独的探索工具

型的探索工具常常包含有 DW2.0 环境下的数据的子集。它很少从 DW2.0 环境中直接复制，甚至连一部分也很少复制。

探索工具中的数据具有最低粒度级。另外，它还常常引入大量的历史数据。这么做的原因在于，为了满足探索分析的需求常常需要数据的细节和历史跨度。

探索工具中的数据结构往往是混合型的。一些数据在磁盘上以表格的形式存储，还有一些数据分布在文件中。这些平铺的文件往往最适合于做统计分析。

探索工具往往包含大量同一种类型的数据。探索工具中数据的种类较少，而数据记录却很多。

7.9 探索型处理数据的来源

探索工具可以从很多地方得到所需的数据资源——例如从整合区、归档区和近线区。图 7-8 显示了探索工具从归档区和近线区获取数据的情况。然而，在 DW2.0 中，整合区是探索工具获取数据的主要来源。

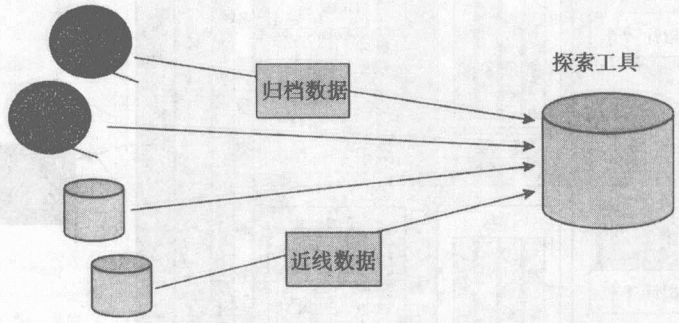


图 7-8 归档数据和近线数据都可以被探索工具使用

有时探索工具也从交互区获取数据。然而，从交互区获取数据有几个注意事项：先要保证交互区的服务不被干扰，即如果探索数据从交互区抽取的话，我们需要特别小心维护该区的性能水平。第二个要注意的是，如果数据从交互区提取进入到探索工具中，我们必须明白这些数据不能是可审查的。举例来讲，一组上午 10:31 分从交互区提取的数据可能在一分钟后就不存在了。如果以上注意事项都考虑到了，那么交互区就可以为探索工具提供合理的数据了。

7.10 更新探索数据

图 7-9 描述了利用 DW2.0 环境数据对探索工具数据的更新。

进入到探索工具的数据的更新周期必须仔细考虑。在 DW2.0 的其他部分，数据的流动会很快，只要出现数据就开始流动。而探索工具的数据只在探索分析师需要时才会流进来。这种需求频率可能是天、周或者是月，这取决于探索分析师的需要。

7.11 基于项目的数据

通常探索工具是基于项目的，也就是说管理层需要对一个具体问题进行研究。收集相关的数据，对数据进行分析，然后把分析结果送至管理层。一旦数据被送至管理层并被研究完，数据要么被丢弃，要么被保存起来。所以基于项目的探索工具并不是一个永久性结构。

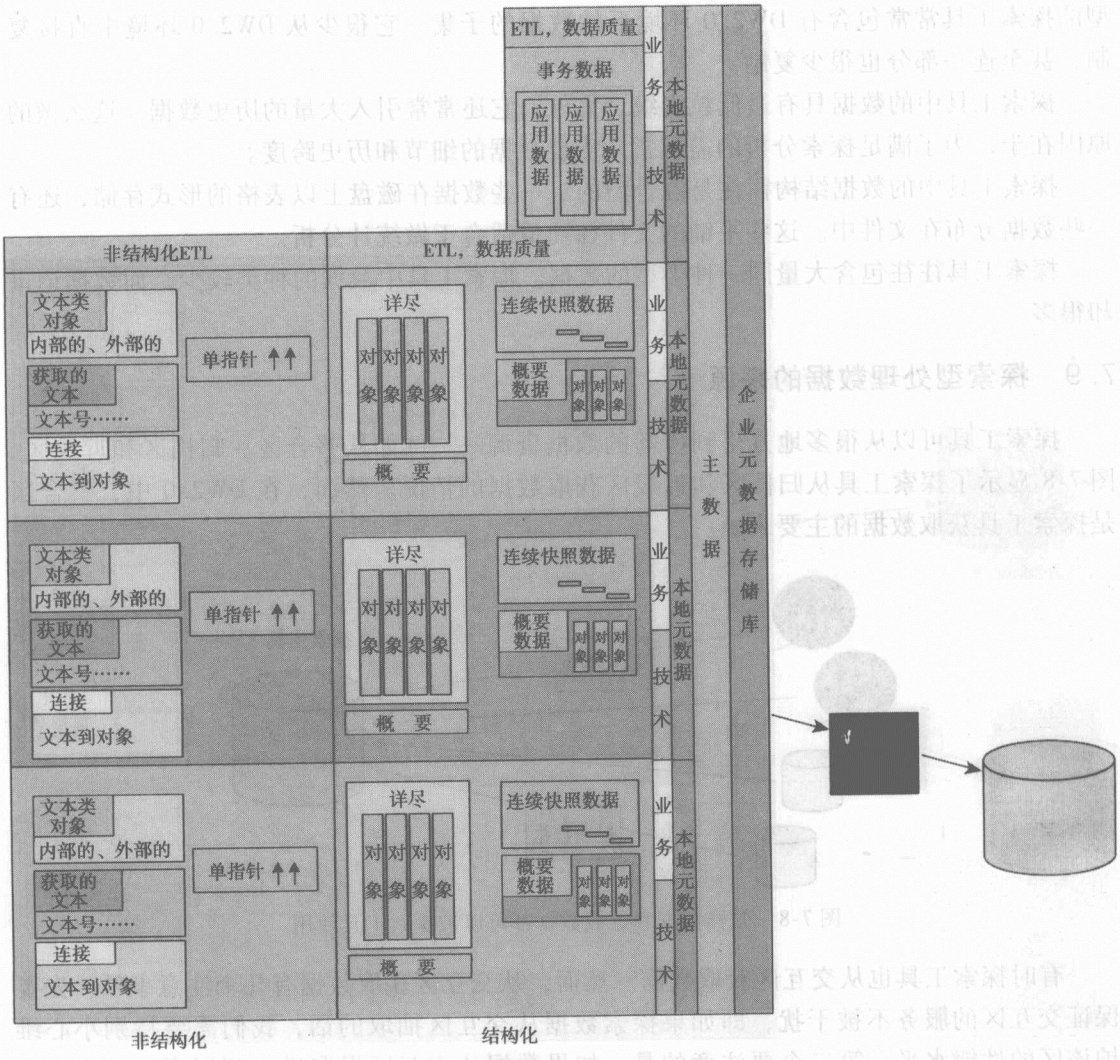


图 7-9 从 DW2.0 环境到探索工具数据的定时更新

然而一些机构却想拥有永久性的探索工具。在这种情况下，当需要进行分析时，探索工具需要随时可用，而其中的细节数据则要经常更新。

图 7-10 表明探索工具中的数据可以是永久的或者临时的。

7.12 数据集市和探索工具

分析员经常认为，探索工具往往就是或相似于数据集市。实际上，数据集市和探索工具有很大的不同。一些主要的区别如下：

- 探索工具拥有细节数据，而数据集市则多为概要或集成数据。
- 探索工具是用来发现知识的，而数据集市则仅仅是传播信息的。
- 探索工具服务于数学工作者，而数据集市则为商业分析员提供帮助。
- 探索工具是基于平铺文件的，而数据集市则是基于 OLAP 的。
- 探索工具经常是临时性的，而数据集市则具有永久性。
- 探索工具依赖于统计软件，而数据集市则往往依赖于商业智能化软件。

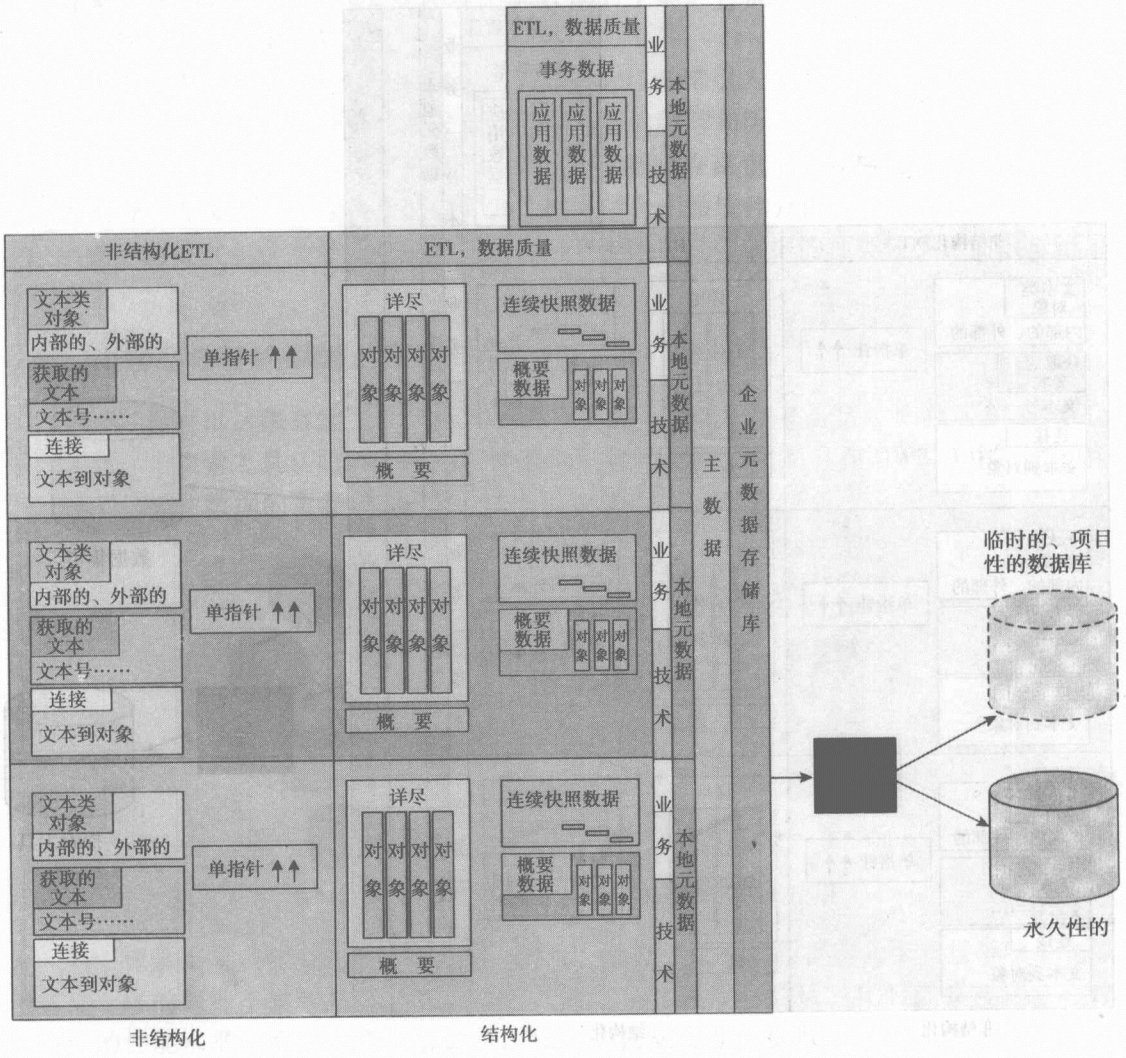


图 7-10 探索工具可以是永久性结构或者临时性结构

以上就是探索工具和数据集市的主要区别。

图 7-11 表明探索工具和数据集市是构架中有很大差异的组成部分。

7.13 数据回流

另一个有趣的问题是让数据从探索工具流回到 DW2.0 环境是否是明智的。事实上这么做是允许的，但是有一些前提条件必须满足。

以下是一些所需的条件：

- 探索工具输出的数据必须能够在整个企业环境的不同地方使用。如果输出数据仅仅被用在一两个地方，那么将它置于 DW2.0 环境下便没有什么意义。
- DW2.0 环境中需要有和探索工具数据相关的数据审核跟踪及计算。
- 如果探索工具中的数据要放入 DW2.0 环境中，并且该探索工具是基于项目的，那么这些数据往往是受限制的一次性提供的数据。换句话说，如果要放入 DW2.0 环

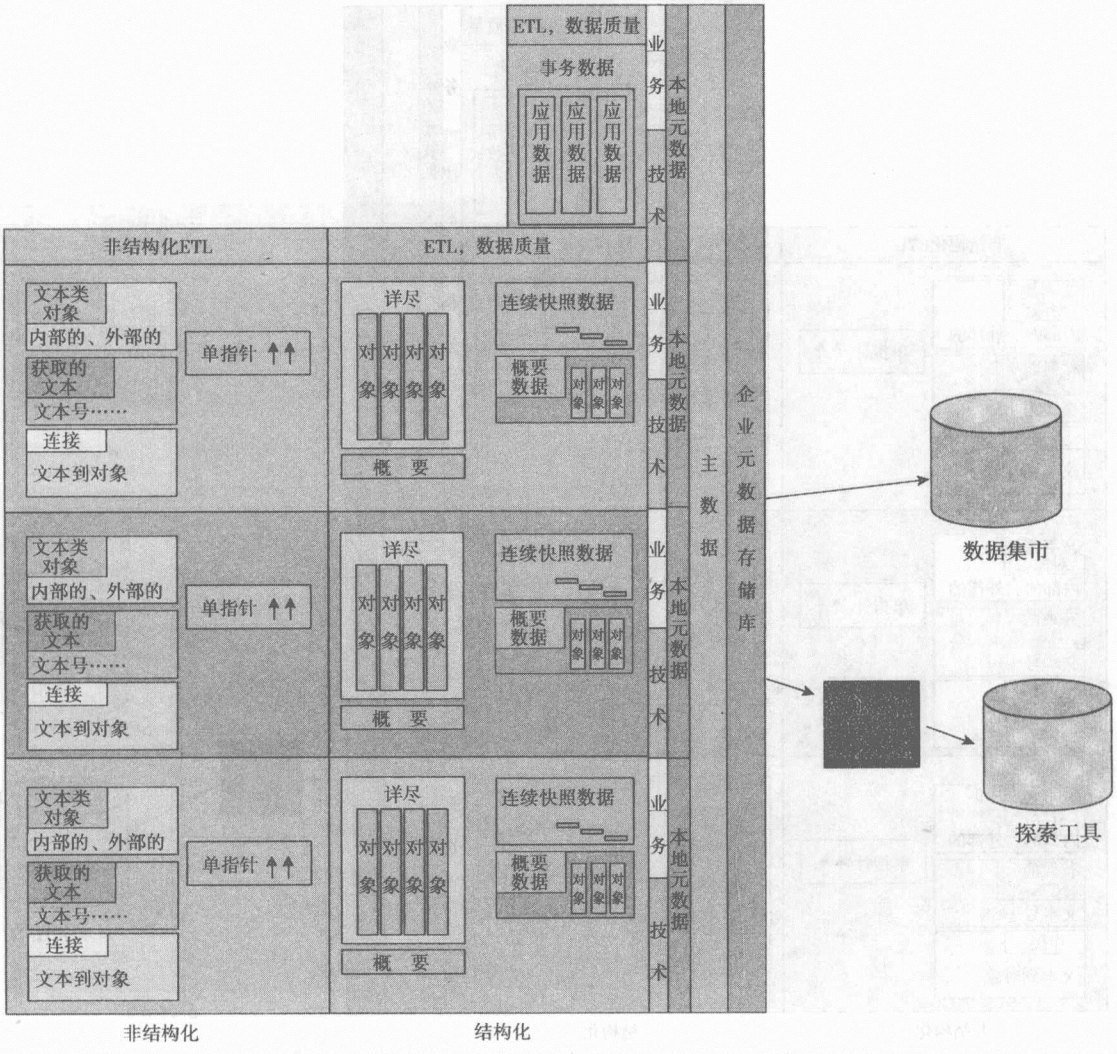


图 7-11 数据仓库和探索工具有很大的不同

境中的数据来自于临时性资源，就不要指望这些资源成为进入 DW2.0 数据仓库的数据的永久提供者。

图 7-12 描述了在正确情况下，潜在的数据从探索工具流向 DW2.0 数据仓库的反馈。

7.14 在内部使用探索数据

探索工具在使用时一定要谨慎。大多数情况下，探索工具提供的分析仅仅被内部使用。这是因为探索工具用到的数据并没有像流入或流过 DW2.0 环境的数据那样经过严格的 ETL 处理。因此，当为审计师和审查员提供报告和数据时，如果使用从探索工具得到的数据，那么便没有意义。相反，只有“正式的”数据才可以被用在正式的报告中。我们需要记住，报告用到的信息最终往往会出现在金融评论甚至新闻中，因此将基于探索工具数据的报告用在公共场合中是很不明智的，原因在于这些报告可能并不是通过适宜的计算得到的，甚至还可能包含错误的数据。

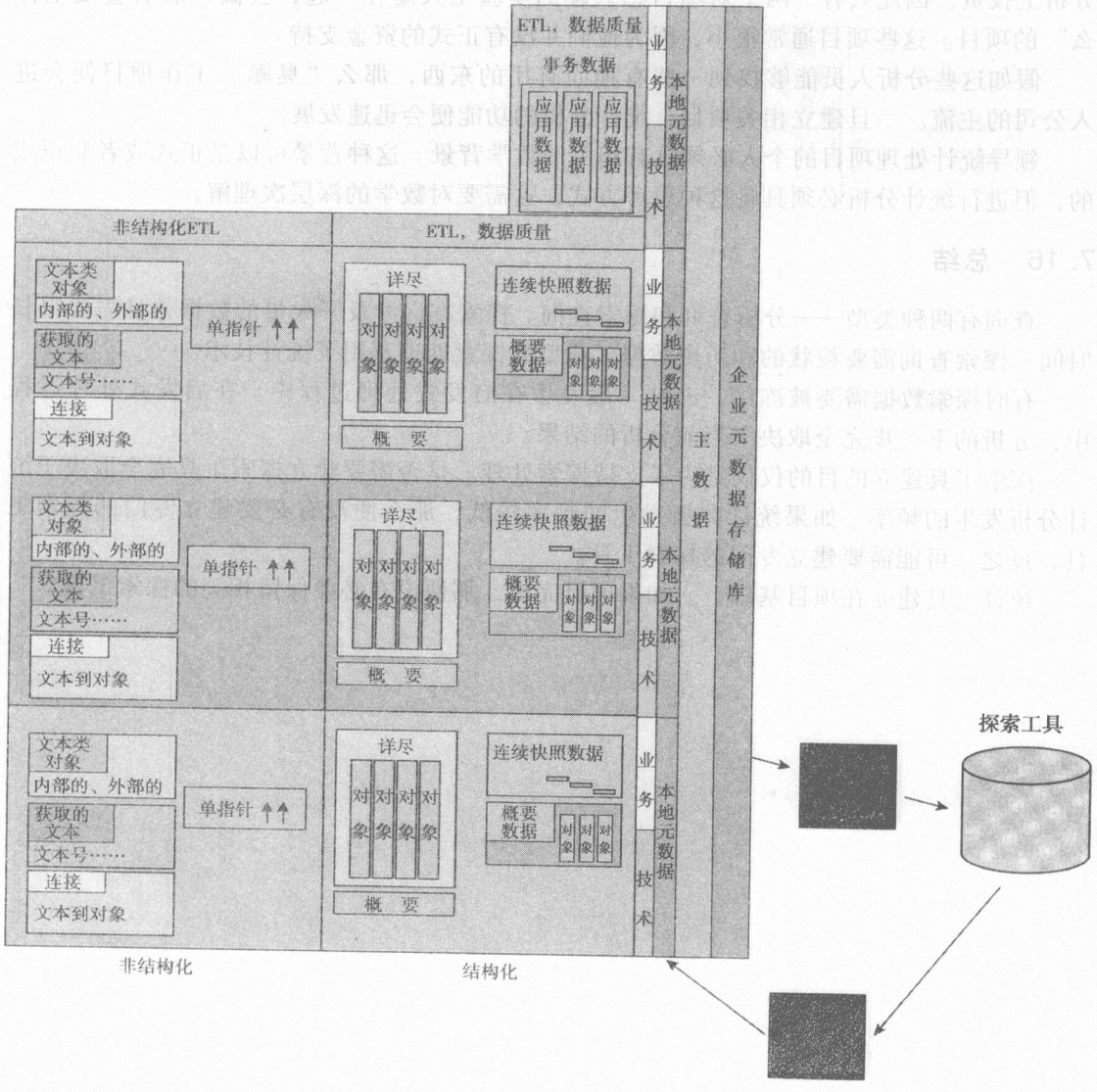


图 7-12 数据有可能从探索工具流入 DW2.0

7.15 企业分析员的观点

很多企业并没有利用它们所拥有的数据的统计处理功能。因此，它们对所拥有的信息资源并没有充分的利用。但是也有一些行业人员很早便发现了信息的价值，并且可以对这些数据进行统计处理。这些典型的行业人员包括保险统计师和研究工程师等。

众所周知，在保险和工程制造部门，统计处理扮演着非常重要的角色。进一步，这种角色很快被行业所认可，并雇用了相关类型的分析人员。

如果要使那些以前没有广泛使用统计分析的企业开始广泛地使用这项技术，那么就需要一些成功的案例说服它们。这些成功案例并不是凭空魔术般生成的，而使用统计分析通常是“臭鼬工厂”计划的结果。因为以前没有成功的案例，所以企业并不愿意在统计

分析上投资。因此只有一两个对项目感兴趣的实验工人凑在一起，去做“看看会发生什么”的项目。这些项目通常很小，因为他们并没有正式的资金支持。

假如这些分析人员能够找到一些有趣而有用的东西，那么“臭鼬”工作项目便会进入公司的主流。一旦建立相关项目，统计分析的功能便会迅速发展。

领导统计处理项目的个人必须具有相关的数学背景。这种背景可以是正式或者非正式的，但进行统计分析必须具备这种思考方式，这需要对数学的深层次理解。

7.16 总结

查询有两种类型——分析查询和探索查询。探索查询涉及了大量的数据并且需要很长时间。探索查询需要粒状的和历史数据。典型的探索处理利用了统计技术。

有时探索数据需要被冻结，冻结一般发生在启发式处理过程中。在启发式处理过程中，分析的下一步完全取决于当前分析的结果。

探索工具建立的目的仅仅是为了支持探索处理。是否需要建立探索工具完全取决于统计分析发生的频率。如果统计分析发生的频率较低，那么便没有必要建立专门的探索工具，反之，可能需要建立专门的探索工具。

统计工具建立在项目基础上。如果项目完成，那便没有必要保留相关的探索工具。

第8章　数据模型与DW2.0

DW2.0是一个非常复杂的“世界”，它涉及了很多不同的方面，所以很容易会纠缠在具体细节中并很快迷失方向。因此，在使用DW2.0时，保持自己的观点非常重要。

8.1　智能路线图

因此，数据模型就成了DW2.0一个必备的组成部分。数据模型在很多方面都扮演着通往DW2.0其他部分的智能路线图的角色。图8-1显示了DW2.0中数据模型的这种角色。

在DW2.0中建立路线图有很多种原因，但最重要的原因或许是建立DW2.0并不是一蹴而就的。相反，它是每次建立一步，通过很长的时间来完成的。另外，它的建立是由很多人而非单个人完成的。

为了协调不同人员的工作以及适应不同类型的用户，非常有必要建立一个路线图——数据模型，数据模型描述了DW2.0各部分如何结合在一起。如果没有数据模型，DW2.0各部分的工作便被割裂开来，从而导致系统的混乱。

8.2　数据模型和企业

数据模型是依据企业本身而建立的，它模拟了企业的各个部分。

图8-2描述了建立在企业上的数据模型。

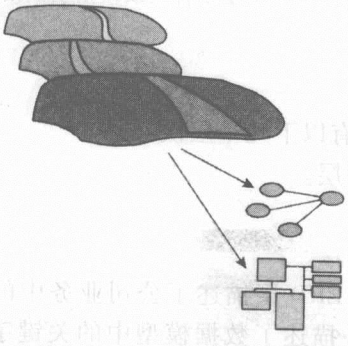


图8-1　数据模型扮演着智能路线图的角色

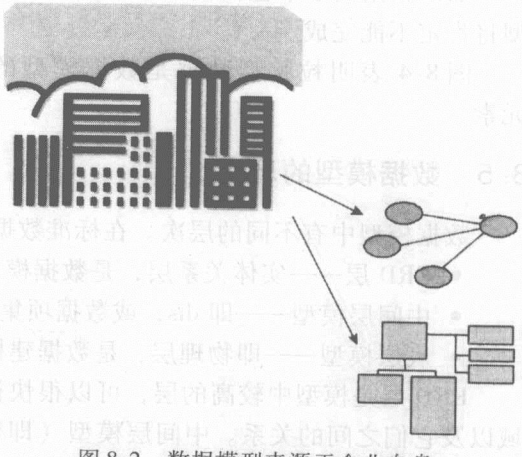


图8-2　数据模型来源于企业自身

8.3　整合范围

建立数据模型的第一步是定义整合范围。整合范围就是描述数据模型中包含什么和不包含什么。整合范围是十分重要的，因为没有它数据模型便会无休止地建立下去，甚至

可能包含宇宙级的数据。当这种现象发生时，数据模型的建立永远都不会停止。

图 8-3 显示了整合范围的定义是建立数据模型的起点。

数据模型建立在企业的数据基础之上。大多数机构都有大量的数据。这样，即使定义了整合范围，如果分析员不够谨慎，数据模型的建立还是会无休止地进行，除非对粒状型数据和概括型或聚合型数据有明确的区别。粒状型数据是指体现最底层意义的数据。一个人的姓名是粒状型数据，生日也是粒状型数据，薪水在一个时段内可以看成是粒状型数据。

图 8-3 建立数据模型的第一步是确定整合范围

概括型数据则是诸如一天的交易量、一个月的收入、一年里企业的员工数、一个季度内的国民生产总值之类的数据。

8.4 区别粒状型数据和概括型数据

关于为什么要区别粒状型数据和概括型数据主要有以下几个原因：

- 概括型数据远远多于粒状型数据。
- 概括型数据变化速度比对其建模过程要快。
- 概括型数据自身携带描述其是如何形成的算法。

如果数据模型中包括概括型数据，那么该模型将肯定不能完成。

图 8-4 表明粒状型数据是数据模型的构造元素。

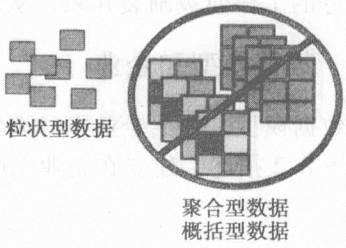


图 8-4 数据模型专注于粒状型数据，而不是聚合型或概括型数据

8.5 数据模型的层次

数据模型中有不同的层次。在标准数据模型中，有以下几个层次：

- ERD 层——实体关系层，是数据模型中的最高层。
- 中间层模型——即 dis，或数据项集。
- 底层模型——即物理层，是数据建模的最底层。

ERD 层是模型中较高的层，可以很快被构建好。ERD 层描述了公司业务中的主要领域以及它们之间的关系。中间层模型（即数据项集）描述了数据模型中的关键字、属性以及细节数据之间的关系。底层模型描述了数据模型的物理特性，例如数据的物理属性、索引、外键，等等。

模型的层次越低，细节层次就越高。而模型的层次越高，模型就越完善。

图 8-5 显示了数据模型的各个层次。

事实上，像 DW2.0 那样拥有不同层次模型的复杂结构在现实中也很常见，并不是一

项新的或者陌生的技术。比如图 8-6 所示的世界地图就是一个很好的例子。

在图 8-6 中，我们可以看到美国地图、得克萨斯州的地图以及得克萨斯州达拉斯市的交通图。每一张地图都和其他地图存在着联系。得克萨斯州可以在美国地图中找到，达拉斯市可以在得克萨斯州的地图中找到，因此它们之间存在着相关性。

每一张地图都有不同的细节层次。在美国地图上可以找到美国的州际高速公路系统，在得克萨斯州的地图上可以找到得克萨斯 285 号线路，而道顿的 Grapevine 路则可以在达拉斯附近找到，因此细节层次在以上的地图中依次递减。

相应地，每幅地图的完整性随着层次的降低而减小。美国地图只能显示美国的情况而不能显示巴西的，得克萨斯州地图只能显示得克萨斯州而不能显示亚利桑那州或者田纳西州的情况，同理达拉斯市地图不能显示桑得森或者德尔里奥的情况。

不同层次的映射结合在一起组成了一个有层次的整体。

同理，如果数据模型结合在一起，那么组成 DW2.0 环境的各种系统便有了新的意义和秩序。图 8-7 显示了数据模型以及它给信息系统带来的秩序。

DW2.0 环境中有很多不同的模型，不能想当然地认为它仅有一个模型。

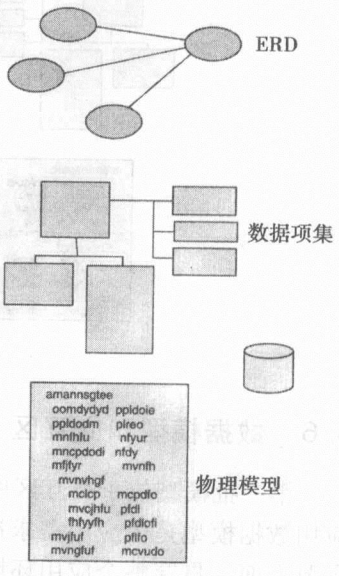


图 8-5 数据模型有不同的层次

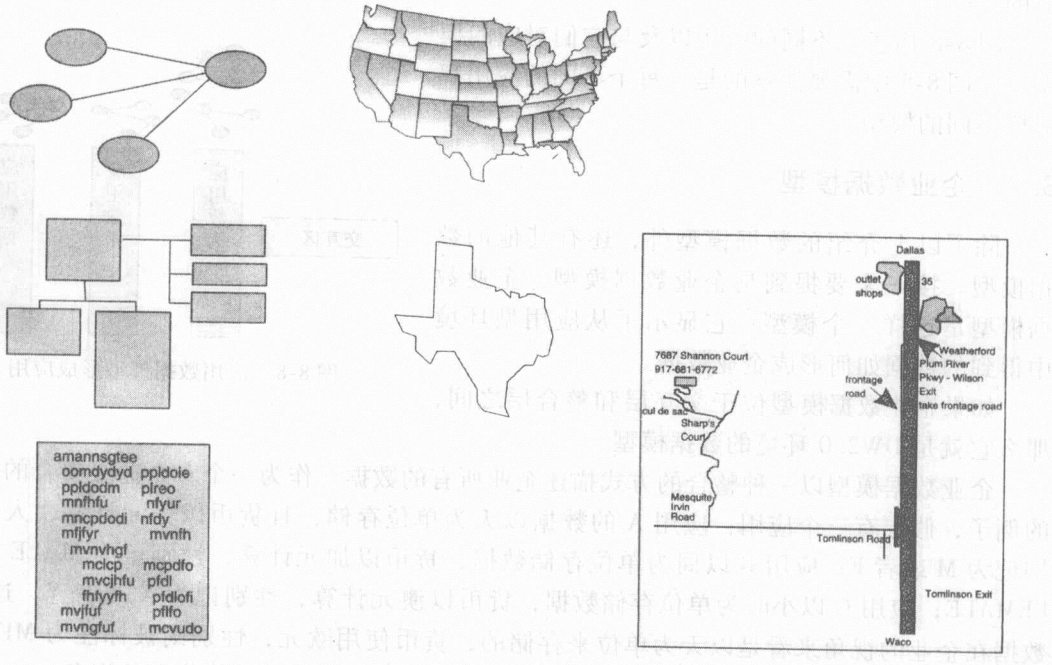


图 8-6 不同层次数据模型之间的关系

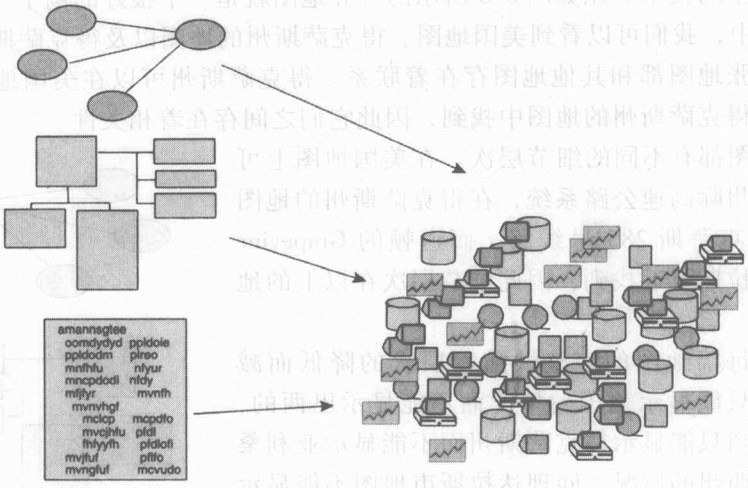


图 8-7 数据模型使 DW2.0 有秩序

8.6 数据模型和交互区

第一批模型位于交互区中的应用当中。通常，每一项应用对应一个单独的数据模型。应用数据模型是由应用需求决定的。对性能的需求是建立应用数据模型需要考虑的一个主要方面。贯穿整个应用环境，特别是那些存在 OLTP 事务的地方，数据模型都是根据性能需求建立的。当数据模型考虑了性能时才会变得合理化，贯穿系统的合理数据流会产生高的性能表现。而这种合理数据流正是由合理化的数据模型将数据放在一起来形成的。

图 8-8 描述了不同的应用以及与它们对应的模型。在图 8-8 中需要注意的是，每个不同的应用都对应不同的模型。

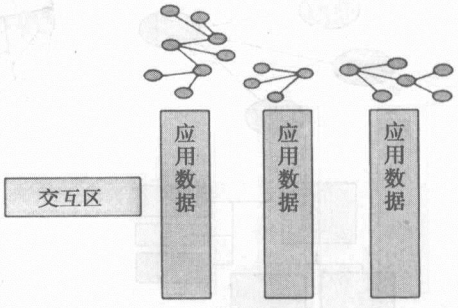


图 8-8 应用数据模型形成应用

8.7 企业数据模型

除了以上介绍的数据模型外，还有其他的数据模型。接下来要提到是企业数据模型。企业数据模型是这样一个模型：它显示了从应用型环境中得到的数据如何形成企业数据。

如果企业数据模型位于交互层和整合层之间，那么它就是 DW2.0 环境的数据模型。

企业数据模型以一种整合的方式描述企业所有的数据。作为一个对企业级整合的需求的例子，假定有三个应用：应用 A 的数据以天为单位存储，且货币以美元计量，人的性别记为 M 或者 F；应用 B 以周为单位存储数据，货币以加元计算，性别记为 MALE 或者 FEMALE；应用 C 以小时为单位存储数据，货币以澳元计算，性别记为 X 或者 Y。这样，数据在企业的视角来看是以天为单位来存储的，货币使用欧元，性别则被标注为 MEN 或者 WOMEN。数据模型真实反映了企业看待信息的视角，是一种整体信息的视角。

8.8 模型转化

图 8-9 表明当数据从应用区或交互区流入整合区时发生的数据的基本转化。值得注意的是，当数据进入整合区时，通常会按对象域存储。

当数据流进入近线区时，数据模型通常不会发生改变。因为近线环境需要尽可能地模仿交互环境，因此近线环境下的数据模型和交互区数据模型完全一样。

图 8-10 表明数据进入近线区时数据模型并没有改变。

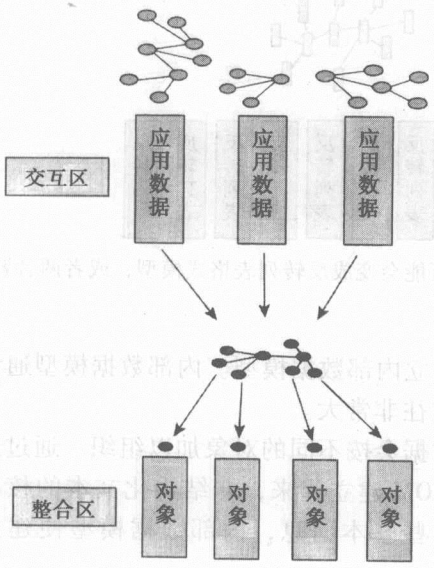


图 8-9 数据发生根本转变，应用数据变为企业数据，企业数据模型决定数据的转换

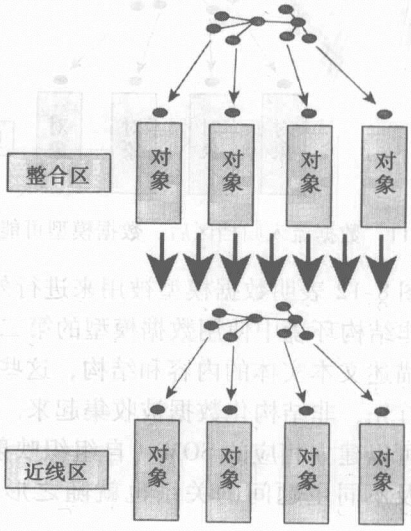


图 8-10 数据从整合区流入近线区，数据模型没有发生变化

最后，数据流入归档区。此时，数据模型可能改变也可能不改变。在一些情况下，数据进入归档区后的状态和在整合区时相同，此时数据模型就没有改变。

但在另外一些情况下，数据流入归档区时会发生根本的改变。在这种情况下，数据流入了一个可称为反转列表格式的地方。当数据流入反转列表格式时，它便被重新安排为一系列简单的列表。

归档分析或许需要这样一个转换，因为它可以使归档环境下的数据更易于查找和分析。当然，归档环境下的数据可以放在企业数据模型格式中或者反转列表格式中。

图 8-11 显示了数据流入归档区的情况。

8.9 数据模型和非结构化数据

对于 DW2.0 的结构化部分，数据模型是适合并且有用的。但在 DW2.0 的非结构化部分，数据模型也常常被用到。不难理解，数据模型对非结构化部分的重要性并没有其对结构化部分的重要性大。

在 DW2.0 的非结构化部分，使用数据模型的第一个地方是外部分类过程。外部分类通常用于对数据进行分组或分类，使这些数据规范化或者合理化。

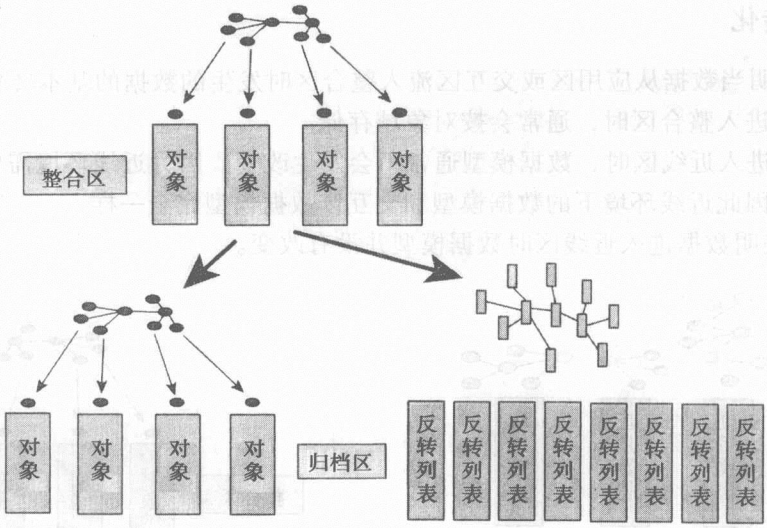


图 8-11 数据流入归档区后，数据模型可能保持原样，可能会变成反转列表格式模型，或者两者皆有

图 8-12 表明数据模型被用来进行外部分类。

非结构环境中使用数据模型的第二个地方是建立内部数据模型。内部数据模型通常被用来描述文本实体的内容和结构，这些文本实体往往非常大。

首先，非结构化数据被收集起来。接着这些数据会按不同的对象加以组织。通过这些对象可以建立相应的 SOM（自组织映射）。一旦 SOM 建立起来，非结构化文本的核心主题以及不同主题间的关系也就随之形成。通过这些基本信息，内部数据模型便建立起来了。

图 8-13 描述了通过非结构化文本实体中的对象建立内部数据模型的过程。

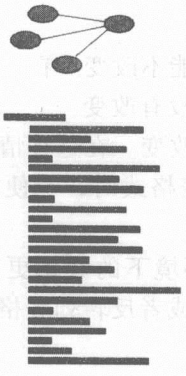


图 8-12 外部数据模型被用来构造非结构环境中的外部分类

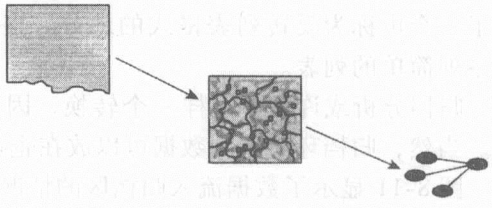


图 8-13 文档可以归约为内部对象，接着内部对象可以建立内部数据模型

8.10 企业用户的观点

企业用户对数据建模过程至关重要。事实上，从正确的角度来看，数据模型是终端用户如何看待 DW2.0 中数据的一种具体化形式。

换种方式说，如果数据模型来源于除终端用户外的其他资源，或者终端用户没有检查并承认数据模型的合理性，那么DW2.0中形成的数据便是不太合适的了。

这也就意味着终端用户需要从开始就要加入，因为在DW2.0建立的开始就要建立数据模型。在开始阶段先建立数据模型，然后在某个时间点展示给终端用户，这样做可能会产生无法正确构建DW2.0的一些主要部分的风险。

某些情况下，在完成数据模型的建立和终端用户输入的记录时，需要有一些非常规范的操作。这些操作包括记录商业人士的言论，并请其签字保证自己说了什么和没说什么。这样做可能是非常必要的，尤其是当终端用户比较健忘或者在一个大型组织中讲给一大群人时。或许在一段时间后就会发现对终端用户说了什么或没说什么做记录稿是非常有用的。

终端用户不必成为数据建模技术的专家。（令人意外的是，一些终端用户如此喜欢建模过程并且通过不停地实践最终成为了数据建模的专家。）相反数据建模过程都是由外部人士来完成，当然他们都是数据建模专家。

一段时间后，商业模型要发生相应的变化。商业人士参与数据模型的修改，就像他当初参与数据模型的建立一样。

8.11 总结

数据模型形成了DW2.0环境的智能线路图。DW2.0规模庞大，结构复杂，需要大量的开发者经过长期的努力才能建立起来。正是数据模型的出现，才使得不同的开发工作能够结合在一起。

数据模型的形成取决于企业的业务需求，它建立在大量的粒状型数据基础上，而非概括型或聚合型数据。

数据模型包括三个层次——ERD层、中间层（或dis层）以及底层（或物理层）。

交互区的形成由应用模型决定，整合区的形成由企业数据模型决定。

一些数据模型可以被用在非结构化数据中。特别是外部分类可以有为它们建立的数据模型。另外，内部数据模型可以根据主题建立，这些主题是根据文本产生的。

第 9 章　监视 DW2.0 环境

DW2.0 环境是复杂和动态的，它的各部分之间存在着复杂的联系。数据要从一个组件流向另一个组件，还要执行事务，并且还进行数据转换等。

在很多方面 DW2.0 环境像是一个黑匣子。数据从一个地方放进去，从另一个地方取出来，而在这之间发生了神秘的事情。不幸的是，如果 DW2.0 环境被看做是一个不透明的黑匣子，我敢打赌随着时间的进行，在黑匣子中发生的事件会变得不顺利起来：数据会收集一些本不该收集的数据，事务响应变得缓慢，数据会放错位置，甚至还有更坏的情况。

因此，DW2.0 环境不应该像是一个黑匣子，这样就需要周期性的查看以便确保 DW2.0 环境以预先期待的方式运行。

9.1　监视 DW2.0 环境

说到底就是强烈推荐要对 DW2.0 环境进行定期的监视。至少，应该在黑匣子中插入一个听诊器以便发现什么正在运行。当 DW2.0 环境或它的某一部件需要调整时，这些调整能提前而非被动地去做。

9.2　事务监视

在 DW2.0 环境中至少需要三种监视。第一种是事务监视，它运行在 DW2.0 的交互区。事务监视用来确保一个良好一致的响应时间。

图 9-1 描述了事务监视。

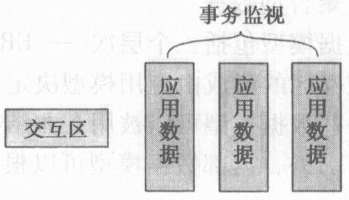


图 9-1　事务监视是 DW2.0 环境中的一种监视

9.3　数据质量监视

DW2.0 环境中需要的第二种监视是对数据质量的 ETL 监视。这种监视专用于核查通过 DW2.0 转换组件的数据的质量。如果低质量的数据被送入 DW2.0，那么至少需要通知数据分析师，使其也意识到这一点。

图 9-2 描述了数据质量监视。

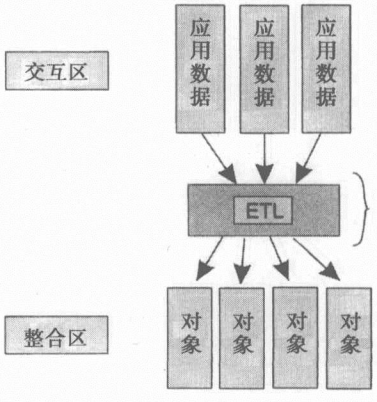


图 9-2　ETL 时刻的数据质量监视也是 DW2.0 环境的一种监视

9.4　数据仓库监视

作为 DW2.0 环境的一部分的第三监视是数据仓库监视。它监视数据仓库中的数据。虽然数据仓库监视用来服务于多个不同的目的，但它主要的目的是测

量数据的使用频率。从数据的使用频率可以确定任一数据是否进入休眠期。对休眠数据的管理是DW2.0环境管理的一个最为重要的方面。图9-3描述了数据仓库监视。

接下来会深入描述DW2.0环境的三种监视。

图9-3 数据仓库监视是DW2.0的重要组成部分

9.5 事务监视——响应时间

事务监视的主要目的是确保良好一致的反应时间。不幸的是，系统运行的许多方面都会对系统性能产生影响。

监视系统性能经常要将响应时间作为一个参考，在2~3秒的范围内的反应时间通常认为可以接受。也可能在一天当中存在一些时间段，在该时间段的响应时间有所延长。但是只要这些时间段比较短且并不频繁，并且响应时间增加得不是太长，那么这个系统就可以被认为是以一种令人满意的方式运行。

通常可接受的响应时间参数定义在服务水平协议中。

事务监视的一些特性和特征包括：

- 事务队列监视：事务队列是事务在执行之前存储的地方。当系统繁忙时，事务会被挂起在事务队列中等待执行。如果系统非常繁忙，那么这种等待就会成为性能的一个最大障碍。
- 应用监视：在电脑中处理事务的应用需要被监视。当一个事务被执行时，它要占用系统资源。这些系统资源用于运行正在执行的事务的代码，而这些资源所被使用的时间长度是系统吞吐量和性能的最重要的度量之一。
- 事务记录监视：完成一个交易所需的记录数也会影响系统性能。一个单独的业务事务经常耗费许多资源。但是最能表现事务处理性能的指标是事务执行所需的记录数。简单地说，需要较少记录的事务执行起来会比必须执行许多记录的事务快得多。

当然，还存在其他许多性能测量指标，但是这些指标是最重要的。

图9-4列举了一些可以使用事务监视来达到最大效益的一些组件。

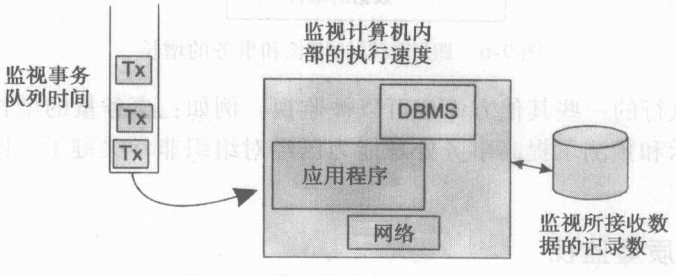


图9-4 事务监视的基本活动

事务监视有许多输出，接下来将讨论其中一些最突出的输出。

9.6 高峰期处理

来自事务监视的一个重要度量标准是系统在高峰期处理中对其所有资源的使用程度。

在每个事务处理环境中，都有不活跃阶段和活跃阶段。活跃阶段就是所谓的“高峰期”。

只要有能力满足所有的处理，系统就会平稳运行。但是在高峰期，如果系统对资源的需求超出可用资源，系统就会变慢，并且在绝大多数情况下会非常明显。因此，每一个组织都应该去监视需要耗费所有可用系统资源的高峰期处理时的资源使用程度。如果在高峰期的资源使用是稳定的，那么就没必要去增加系统容量。如果或当高峰期处理对资源需求持续增长时，这就提示了需要更多的系统资源，或是需要资源的重新分配。

图 9-5 描述了高峰期资源的变化情况。

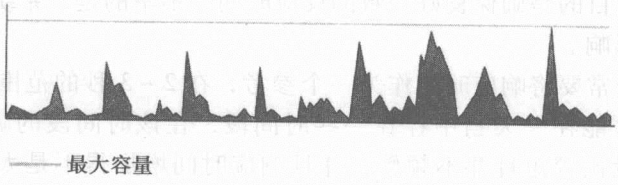


图 9-5 亟待解决的问题——什么时候容易使用达到最大值和达到那一点时将有什么后果

另一个典型地用于事务监视记录的重要参数是系统的增长率。可以随着时间被记录的系统增长的典型指标是系统中的事务数和数据量。

事务的数目是系统增长和容量消耗的速率的一个良好指标。通过推测和设计一个系统处理的事务的数量，系统分析员就能确定什么时候需要进行硬件升级。其目的是预测什么时候需要进行升级和确保在性能问题开始之前组织可以以主动的方式进行响应。一成不变地以被动的方式运行意味着组织将承受周期性的“瘫痪”。由于对公司运行的负面影响，瘫痪已经造成了大量痛苦。

图 9-6 描述了跟踪随时间变化的事务量和数据量的增长的典型结果。

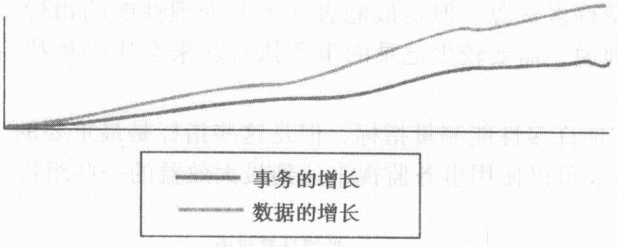


图 9-6 跟踪数据的增长和事务的增长

事务处理的执行的一些其他方面也可以被监视。例如：事务量的增长和相关响应时间降低的叠交就显示和预测了提高事务处理能力已经对组织非常关键了。图 9-7 描述了这种比较性的性能测定。

9.7 ETL 数据质量监视

当数据从 DW2.0 的一个区流入另一个区，或数据最初进入系统时，ETL 数据质量监视就会检查数据。ETL 数据质量监视的目的是为了评价数据被转化时数据的质量。

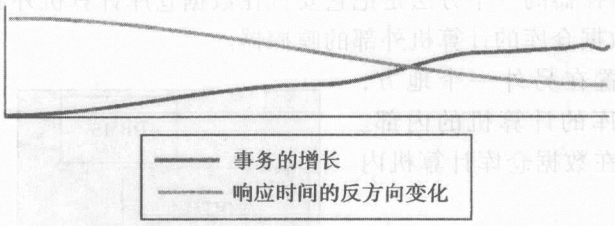


图 9-7 跟踪事务的增长及与此匹配的响应时间的反方向变化

ETL 数据监视查看数据的很多方面，它将检查如下内容：

- 数域：假设性别被定义为："M/F"，如果性别的数据以"MALE"的格式录入，那么 ETL 数据质量监视会将其记录为错误。
- 不匹配的外键：例如，如果数据中存在对"John Jones"的引用，但顾客数据库中没有 John Jones，那么便被认为是外键缺损或未匹配。
- 边界范围：顾客的正常年龄在 15 岁到 80 岁之间。如果进入系统的一个顾客的年龄属性是 234 岁，这显然是一个年龄范围数据质量问题，需要被检测到并报告。
- 空值：指定的每一个数据键都应该出现。如果一个输入数据仓库中的一个记录没有键，那么它必须被检测到并报告。
- 被损坏特征：一个被拼成"Mar［++*］"的名字也可以进入 ETL。除非一个人有一个不寻常的名字，例如以前一个艺术家叫做王子，否则这很有可能是数据中的一个质量误差问题。

还有其他一些需要 ETL 数据质量监视来检测和报告数据质量问题情况的例子。

最为有趣的一个关于数据质量的问题是：一旦检测到一个错误情况，该做哪些处理。一种选择是丢弃这个数据，但是通常情况下这是一个糟糕的选择，因为：

- 被丢弃的这条记录的其他部分可能非常好。
- 需要一些更正方法。人工更正应该是最后的选择，因为人工更正大量不正确的数据需要非常多的时间，会严重拖延项目的进度。

另一个选择是生成缺省数据。虽然也非常有效，但被认定为不正确的数据在系统中再也不存在了。还有另外一个选择就是让坏的数据进入系统，并将其标注为错误的。标注错误数据就是在警告终端用户这个数据存在问题。

图 9-8 描述了 ETL 数据质量监视的位置和角色。

9.8 数据仓库监视工具

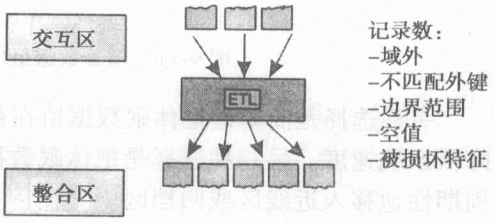


图 9-8 数据质量监视

数据仓库监视是监视数据仓库中什么数据正在被使用和什么数据没有被使用的一个软件工具。如果一组数据相当长的时间没有被使用，那么它就被认定是"休眠的"。好的数据仓库监视应设计为能够检测和报告休眠数据的。

数据仓库中数据监视的一般方式是通过截取提交给数据仓库系统的 SQL 代码。通过收集输入系统的 SQL，分析人员就能确定在数据仓库中什么数据正在被访问，什么数据没有被访问。通常，SQL 是由"嗅探"通信线

路来截取的。安排嗅探器的一个方法是把它安置在数据仓库计算机外部。图 9-9 描述了一个放置在用于管理数据仓库的计算机外部的嗅探器。

嗅探器可以安置在另外一个地方，即用于管理数据仓库的计算机的内部。图 9-10 描述了安置在数据仓库计算机内部的嗅探器的位置。

通常，从管理数据仓库的计算机的外部来嗅探 SQL 代码更为有效。这是因为，当嗅探被允许成为数据仓库 DBMS 的一部分或直接作用于数据仓库 DBMS 时，它的开销会是一个很大的因素。

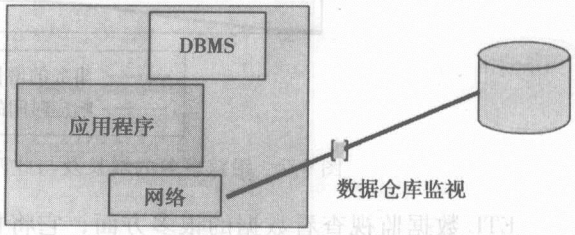


图 9-9 放置在网络内计算机外部的数据仓库监视

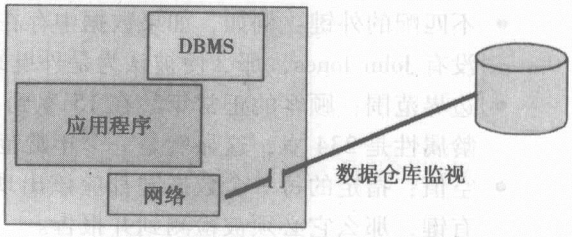


图 9-10 放置于计算机内部的数据仓库监视

9.9 休眠数据

数据仓库中需要一个静态数据监视器有许多原因。主要的原因是当数据进入休眠时，它需要被移入备用存储器中。备用存储器比起高性能的硬盘存储来说要便宜得多。除此之外，休眠数据会阻塞高性能硬盘存储的“动脉”。

把静态数据移入备用存储模式有两个好的理由：

- 省钱——潜在地省了大笔的钱。
- 提高性能。

休眠数据是悄悄进入系统的。图 9-11 描述了休眠数据在数据仓库中是如何增长的。

新建立和实施的数据仓库一般不会包含大量的数据，因此也不会包含很多休眠数据。随着数据仓库中数据量的增长，休眠数据所占的百分比也在增大。当数据仓库中存在非常多的数据时，相应地也会有非常多的数据进入休眠状态。

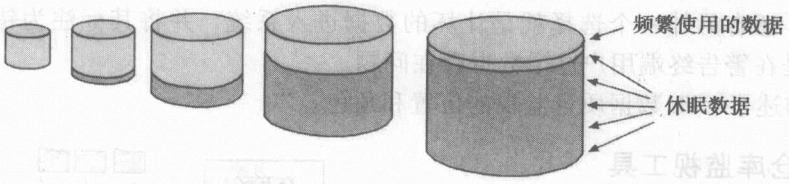


图 9-11 随着数据量增大，休眠数据所占比例也在增加

一种选择是简单地把休眠数据留在数据仓库中。但是这样做成本较高，而且会大幅地降低系统速度。另一种选择是把休眠数据移入线性区或归档区。图 9-12 描述了休眠数据周期性地移入近线区或归档区。

数据仓库监视用于报告什么时候数据进入休眠状态。

9.10 企业用户的观点

数据监视是一项技术实践，因此企业用户不直接参与监视。然而，这些企业用户肯定

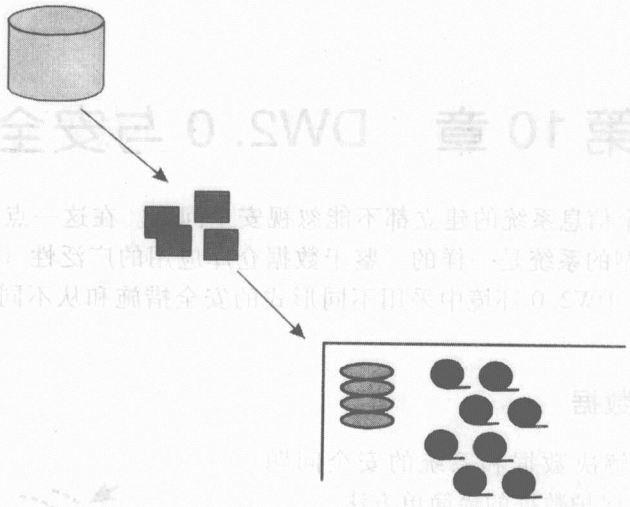


图 9-12 一个亟待解决的问题——什么样的数据需要放置在近线区中

要看见监视结果。

打个比方，当企业用户看到油量警示灯亮起来后，他／她就会开进加油站加油，半个小时后再上路。这时候在路上行驶，油量警示灯就不会再闪烁了。

监视 DW2.0 环境也同样如此。在 DW2.0 环境中企业用户注意到了性能的下降，或者终端用户注意到在一个查询中会返回大量数据。此时，商业人士会同数据构架师讨论这一症状。数据构架师就会利用监视，并解决问题。

值得注意的是使用监视和解决问题可不是一件轻而易举且短时间内能完成的事情。即使在最好的情况下，在注意到问题的症状和解决问题之间也存在一段相当长的时间。

9.11 总结

在 DW2.0 中所需的三种监视为：

- 事务监视
- 数据质量监视
- 数据仓库监视

事务监视放置在交互区，并着重于事务响应时间和性能计划。事务监视尤其关注在高峰处理期发生的数据仓库活动。事务监视需要检测工作量、队列长度和资源利用情况。

数据质量监视着重于监视当数据从 DW2.0 数据仓库环境的一部分进入另一部分时的数域和数据范围。

数据仓库监视主要关注 DW2.0 数据仓库的整合区并解决休眠数据。它观察数据并确定哪些数据被使用，哪些没有被使用。

最好的数据仓库监视是那些在数据仓库 DBMS 外面运行的数据仓库监视。对于监视在数据仓库内部进行的活动，SQL 嗅探的使用是最不冒失且最不耗时的技术。

第 10 章　DW2.0 与安全

当前，任何一个信息系统的建立都不能忽视安全问题。在这一点上，新一代 DW2.0 数据仓库跟其他类型的系统是一样的。鉴于数据仓库应用的广泛性（包括交易处理系统和归档系统等），在 DW2.0 环境中采用不同形式的安全措施和从不同角度解决安全问题就不足为奇了。

10.1　保护访问数据

有许多方法来解决数据和系统的安全问题。图 10-1 提出了一个保护数据的最简单方法。

图 10-1 所描述的屏障在一定程度上可以防止未经同意或未经授权的人们访问数据。这些屏障有多种形式，例如密码、特殊交易和软件的干预。

屏障是有益于整个 DW2.0 环境的：从交互区到整合区、近线区和归档区。

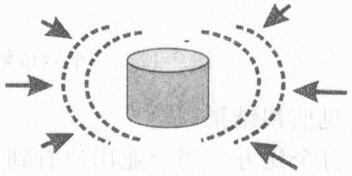


图 10-1　一种保护数据的方式是必须非常小心谁曾经访问过它

10.2　加密技术

还有其他保护数据的方法。图 10-2 描述了另外一种方法。

当数据被加密时，它以一种不同于其初始格式的格式重写。虽然任何人都可以访问加密的数据，但只有那些知道如何解密数据的人才可以理解它真正的含义。通过加密保护数据并不是要保护它不被访问，而是通过将解密机制限制为只有授权用户才能得到来保护数据。在 DW2.0 环境中，加密技术下在某些情况下是有用的，但由于它存在一些弊端，因此使用它必须非常小心。

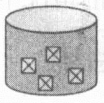


图 10-2　另一种保护数据的方式是加密数据

数据的访问保护和数据加密，这两项安全技术以不同的形式、不同的方面应用于整个 DW2.0 环境。

10.3　缺点

这两种类型的安全技术都有缺点。保护数据不被非授权地访问需要一个自有的技术基础设施。因此，需要一定的管理成本来维持保护性基础设施的更新。

数据加密也有明确的相关成本。当数据被加密时，就会关闭数据仓库系统的一些重要部分。加密的数据不能被有效索引；访问数据时，人们在执行查询前必须先对查询中的参数进行加密；而且，它不能用于逻辑或数值的计算或比较。

简而言之，数据的加密技术有许多缺点。图 10-3 描述了一个这样的缺点。

由于这两种类型的安全技术都存在缺点，因此没有一种类型能够单独满足 DW2.0 数

据仓库的需要。相反，在DW2.0环境中存在一类混合的安全技术。

10.4 防火墙

也许最著名的安全类型就是防火墙了。防火墙用于连接外部世界与公司的内部系统的网络中。图10-4描绘了一个正在管理控制从Internet进入内部网络的事务处理的防火墙。

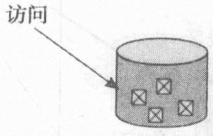


图10-3 当数据被加密时，访问数据的人必须意识到在进行检索之间对数据的加密

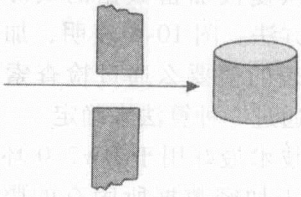


图10-4 对于在线用户的安全，防火墙是一种基本形式的保护方式

值得指出的是，防火墙保护仅仅应用于对交互区的访问，因为交互区是DW2.0架构中活动的事务处理进行的地方。

10.5 使数据脱机

图10-5表明只有交互区与Internet环境有一个接口。

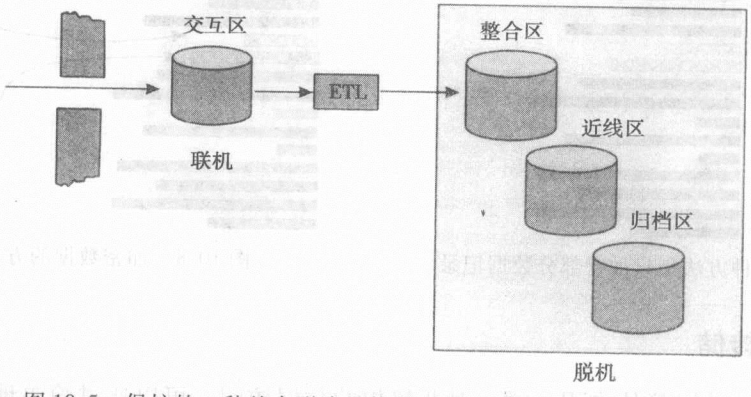


图10-5 保护的一种基本形式是使DW2.0中的非交互式数据脱机

该图表明DW2.0数据仓库中的整合区、近线区和归档区必须不能与因特网有直接的接口。完全将这些数据仓库的关键区域与因特网隔离，意味着即使是最好的黑客也不能得到数据库中的数据，只有交互区中的数据处于风险中。这是一个非常简单但非常有效的安全措施。

允许进入整合区的交互区内数据必须首先经过DW2.0的ETL接口的处理才能进入整合区。在整合区、近线区和归档区中的数据仍然能够被访问；然而，访问它们需要脱机处理或被授权访问机构的内部网络。

图10-6表明DW2.0的数据仍然可以脱机访问。

DW2.0环境中的数据实际上可以被加密。然而，由于对大量的数据进行加密既不实际也不十分有效，所以通常只对一些选定的部分数据进行加密。

10.6 限制性加密

图10-7描述的加密只涉及记录中数据的几个字段。尽管对加密数据建立索引在技术上是可行的，但数据加密后再想获取它并不容易。通过仅对记录中数据的一小部分进行加密，可最小化加密所带来的弊端。

为了恢复被加密数据的实际值，就需要一些方法。图10-8表明，加密数据实际所代表的值要么通过检查索引来确定，要么通过一种算法来确定。

加密技术最好用于DW2.0环境的整合区。因为加密数据所固有的弊端，如果要将其应用在所有地方，那么就应该有节制地使用。当加密技术被用于整合区时，应特别注意这是否会给性能带来消极甚至严重的影响。而加密技术是否应该用于归档区仍然是值得商榷的。

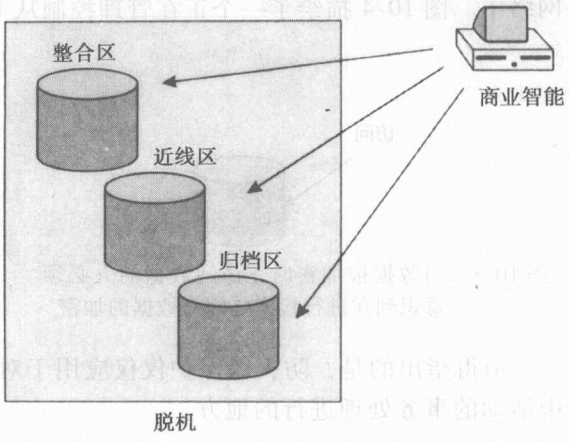


图10-6 通过一个单独的网络访问脱机数据

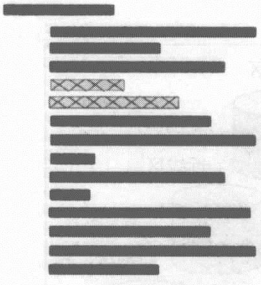


图10-7 一种方法是只加密部分数据记录

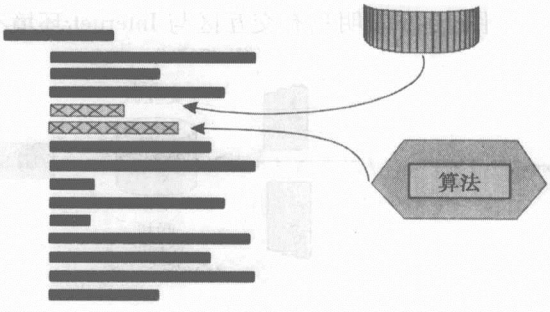


图10-8 加密数据的方式访问

10.7 直接转储

跟一个有效的屏障的作用一样，对非授权的数据访问，可以通过简单地进行一个数据的系统转储，然后手动读取数据来绕过安全保护问题。图10-9描述了一个DW2.0数据库的系统转储。

当进行系统数据转储之后，除了加密以外，数据不受其他任何保护，此时可以手工读取数据，如图10-10所示。

大多数人并不会手工读取被转储的数据，无论是为了娱乐还是为了工作。如果要想完整地读取一个被转储的数据集，还要遵守一套晦涩难解的规则。然而，被转储的数据库中，文本数据可以被很容易地读取，不需要专门的技术或工具。因此，如果仅是要从转储数据中挑选出文本，那么一个非技术人员也可以读取转储数据并做到这一点。

因为读取转储数据绕过所有其他形式的保护，因此最好有某种形式的加密保护。这种保护尤其适用于DW2.0环境中的整合区。

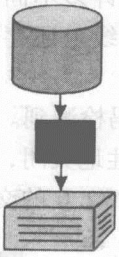


图 10-9 绕过所有安全措施的一个方法是进行一个系统转储

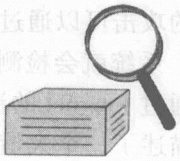


图 10-10 系统转储之后，只能手工读取数据

10.8 数据仓库监视

数据仓库监视是 DW2.0 环境的一个标准建议。在第 9 章中描述了推荐监视的原因。但还有另一个更好的理由来使用数据仓库监视，即确定谁正在寻找什么样的数据。确定谁正在提交查询，查询什么数据库，这些是数据仓库监视可以做的最有用的事情。数据仓库监视和事务监视这两者都产生事务日志。事务日志就是一个能被监视监测到的活动的记录。

图 10-11 描述了一个被监视的 DW2.0 环境，其结果是建立一个事务日志。当监视生成日志后，读取日志是一件相当简单的事情。分析师通过读取数据仓库监视的事务日志，可以知道谁正在查询什么数据。

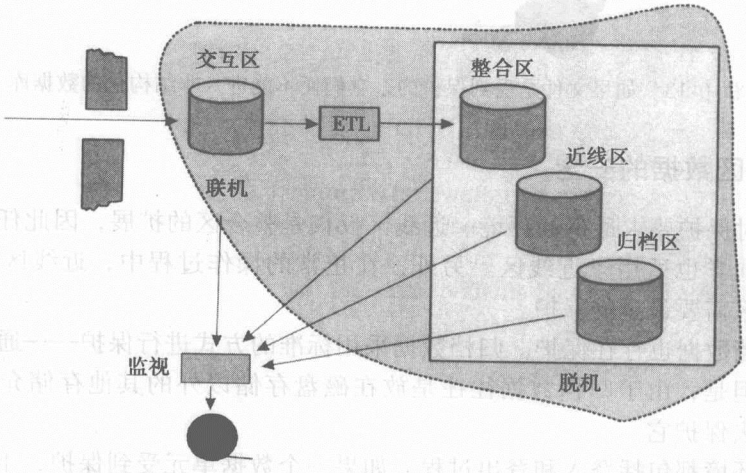


图 10-11 在访问日志中可以看出谁正在寻找什么样的数据

这种形式的安全是消极的，因为它并不能防止对数据仓库数据库的未经授权的访问。但是它仍然是一种安全技术，如果有未经授权的访问，它们都会出现在日志记录中。

10.9 检测攻击

对来自 DW2.0 环境以外的攻击，在它成为一个严重问题之前，或最好在它发生之前，进行检测是另一种很好的保护数据仓库的方法。迅速确定无效或未经授权的访问可以使机构能检测到并停止一个攻击。例如，有时候一个系统在不知道密码的情况下想要进入

另一个系统。解决这个问题的一个有效办法就是在系统内设置许多不同的密码以防止未经授权的访问。一旦攻击产生了能够进入系统的密码，攻击路线上还存储了其他好的密码供今后使用。

这种类型的攻击可以通过记录发送给系统的不被接受的密码检测到。如果突然出现大量无效的密码，系统就会检测到一个攻击。然后系统可以选择性地关闭，直至攻击结束。

随着事务通过 Internet 的流入，这种攻击最有可能发生在 DW2.0 的交互区。

图 10-12 描述了一个大量密码涌入的攻击。

DW2.0 数据仓库的非结构化部分的安全保护模仿了结构化方面的安全保护，但也夹杂了一些增加的部分。如果传入的文件是受保护的并且系统能检测到这种保护，那么这些文件根本不会被带入 DW2.0 环境中。这就确保了对外部保护机制的承认和接受，并确保了拥有其他保护的数据绝不会进入 DW2.0 的非结构化部分。

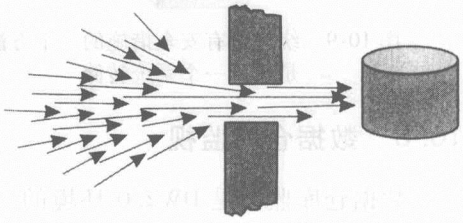


图 10-12 如果太多不成功的访问迅速、连续地要求通过防火墙，系统便可感知到一个攻击可能会发生

图 10-13 表明在进入 DW2.0 时对外部的安全的承认和接受。

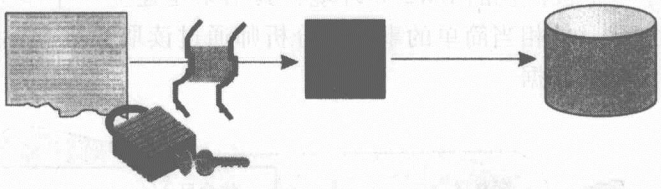


图 10-13 如果文件是受到保护的，它们便不能进入非结构化的数据库

10.10 近线区数据的安全

近线区是对保护要求最少的环境。近线区仅仅是整合区的扩展，因此任何适用于整合区的安全措施同样也适用于近线区。另外，在正常的操作过程中，近线区不能访问它自身；因此近线区需要最少的保护。

最后，归档数据也存在保护。归档数据采用标准的方式进行保护——通过授权访问以及加密技术。但是，由于归档数据往往是放在磁盘存储以外的其他存储介质中，因此有更多的可能性去保护它。

许多归档环境都包括登入和登出过程。如果一个数据单元受到保护，不允许对数据进行登出是一个很好的方法，这就对数据增加了一个额外的保护层。

图 10-14 表明，归档数据提供了更多的安全的机会。

10.11 企业用户的观点

从企业用户的角度出发，安全是绝对必要的。企业用户认为不管怎样，反正数据理所应当应该是安全的。大多数终端用户并不是安全方面的专家。他们只是希望安全是存在的，但是不希望看到安全的任何具体形式。

这就是一个问题，某些类型的安全是非常繁琐并且影响到系统的日常使用。这就像要

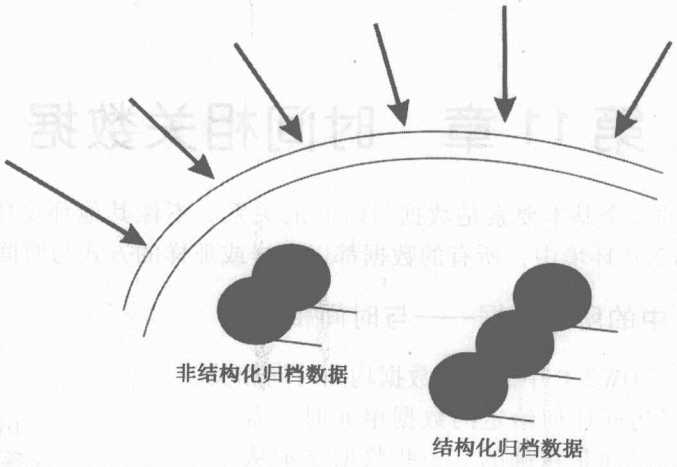


图 10-14 归档环境通常可以有自己单独的安全等级

进入一个国家时等待海关安检，人们只期望稍微等一会儿。虽然有时候人们会多等一会儿，但不希望每次入境时都要等待数小时。

对于 DW2.0 的安全问题人们持有相同的态度。人们预算出一定的总开销，但是这种开销用最好不要一直超出预算。

10.12 总结

对于整个 DW2.0 环境来说安全是必需的，DW2.0 环境的多样性和复杂性要求在整个环境中存在多种不同形式的安全。

有两种基本类型的安全：（1）安全屏障，对于未经授权的访问，在其前面设置一定的障碍；（2）数据加密安全，任何人都能访问数据，但只有经过授权的人才能理解数据的真正含义。

还有一种消极的安全，这种方式并不试图阻止人们访问数据，但使用一个日志记录保存所有被访问的数据。在检测一个未经授权的攻击发生后，这种消极的监视会报告都有什么数据被访问，被谁访问。

原始数据转储是违背安全措施的一种形式。当原始数据被下载时，它绕开所有的保护措施，并在一个原始的状态下检查数据。

另外一种用来保护数据的技术是尽可能多地使数据脱机，这可以防止网上黑客对数据的访问。

另一种形式的安全是攻击监测，它查看是否存在对数据的数目异常的未经授权的访问。

第 11 章　时间相关数据

DW2.0 环境的一个基本要素是数据与时间的关系。不像其他环境中数据与时间是无关的那样，在 DW2.0 环境中，所有的数据都以这样或那样的方式与时间相关。

11.1　DW2.0 中的所有数据——与时间相关

图 11-1 显示了 DW2.0 中的所有数据均与时间相关。

这就意味着当访问任何给定的数据单元时，需要知道数据在什么时间是精确的。一些数据表示从 1995 开始的信息。另外一些数据表示从 1 月开始的信息。还有一些其他的数据表示从今天早上开始的信息。

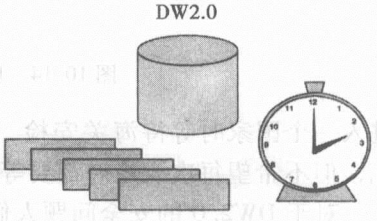


图 11-1　DW2.0 中的所有数据以某种方式与时间相关联

在 DW2.0 中，不论是明确提出还是暗含，所有数据均有一个描述了它的精确性和相关性的相关时间。通常，用于这种描述的记录级别上的一种数据结构如图 11-2 所示。

在图 11-2 中有两种记录类型。一种记录类型用来建立数据在某一时刻的快照。这种记录类型——如左所示——的主键结构中包含日期和时间信息。另外一种类型如右图所示。这种类型含有一个开始日期和一个结束日期，这表示其代表了一个时间块而不是时间点。

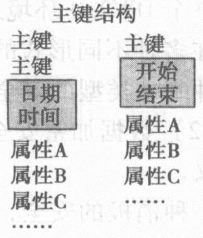


图 11-2　时间是数据结构的一部分

注意在两种情况下，时间元素都是主键结构的一部分。主键是复合键，而时间部分是复合键中的一个组成部分。

11.2　交互区中的时间相关性

在交互区中，数据的时间相关性有些不同。在该区中，数据值被假定为当前访问时间的值。例如，假设你去银行询问你某个账户的余额，那么它返回的时间值被认为在访问时刻是准确的。如果银行工作人员告诉你账户中还有 3971 美元时，那么这个值是累计到当前访问时间的值，考虑到了账户中所有的收入和支出记录。

这样，因为交互区数据用来表示访问时刻的精确值，所以交互区数据中不包含日期信息。图 11-3 显示了正在发生的银行交易，其使用了交互的、实时的数据。

图 11-3　交互区中，数据在使用时刻是正确的

但是在 DW2.0 的其他所有区中——整合区、近线区、

归档区，数据都有一个与之明确关联的时间。

11.3 DW2.0 其他部分中的数据相关

图 11-4 显示了在整合区、近线区和归档区的每一个记录均表示一个时间点或一个时间段。

这种数据与时间相关的概念产生了一些完全不同的处理方式。在交互环境中，完成数据更新。在这种情况下，数据的更新是指数据值的实际改变。图 11-5 显示了在交互环境中银行事务的完成以及数据值的改变。

在上午 10 点 31 分，账户中有 2000 美元。随后产生一笔存入 500 美元的交易。该交易在数据库中的数据上执行，然后在 10 点 32 分时银行账户余额更新为 2500 美元。数据值因为事务的产生而发生了改变。

图 11-4 在 DW2.0 的其他任意区域中，每条记录描述了一个时间段

11.4 整合区中的事务处理

让我们考虑整合区中的一个相似的情况。在上午 10 点 31 分时，整合区数据库中有一个 2000 美元的记录。然后执行一个交易，在上午 10 点 32 分一个新的记录被放到数据库中。这样在数据库中有两个不同的记录，分别显示了不同时间下的不同数据。

图 11-6 显示了整合区中事务的执行。

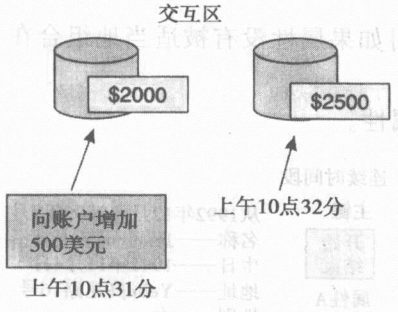


图 11-5 在交互区，因为交互行为数据值被改变

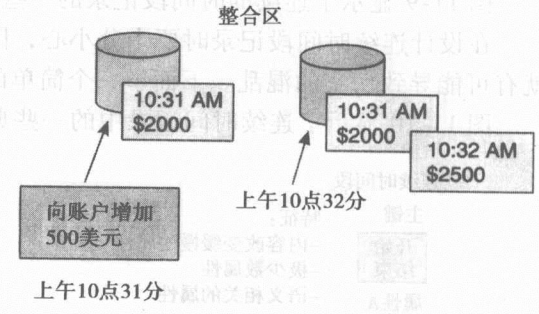


图 11-6 在整合区，所有交互历史记录被保留

图 11-5 和图 11-6 中的不同数据清晰地表明：因为数据和时间的不同关联方式，不同环境下数据库中的数据内容完全不同。

这些不同的数据类型有相应的术语。图 11-7 显示了这些术语。

当仅仅有一个时间点时，数据称为离散数据。当含有起始时间和终止时间时，数据称为连续时间段数据。这两种类型的数据有非常不同的特点。

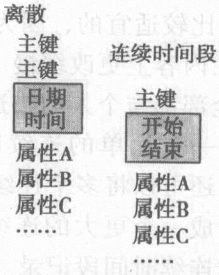


图 11-7 两种时间变化数据的一般形式

11.5 离散数据

离散数据对于大量的快速变化的变量很适用，例如道琼斯工业指数。事实上，道琼斯工业指数是在每天结束时计算，而不在其统计的股票被买

入或卖出后立即计算。被一系列离散的时间快照获取的变量包括了在同一时刻被度量的变量，除了这一点相同外，并没有其他什么能将数据属性与离散记录从语义上联系起来。

图11-8显示了离散结构的数据的一些特征。

11.6 连续时间段数据

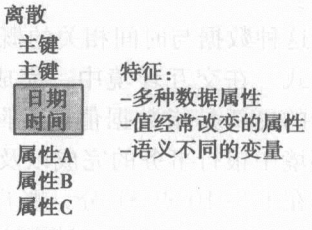


图11-8 离散时间变化数据的一些特征

连续时间段数据有一个不同的特征集。典型地，记录中的连续时间段数据仅有很少的变量，并且这些变量并不经常改变。造成这种特性的原因是每次有一个值变化时，就会重新写入一个新的连续时间段记录。例如，假设一个连续时间段记录包含以下的属性：名称、地址、性别、电话号码。每次这些属性中的一个值发生变化时，就必须写入一个新的记录。名字仅仅是在一个女人结婚或离婚的时候才会发生改变，并不经常改变。地址会改变较为频繁，或许每2～3年就会改变一次。性别对大多数人来说从不会改变。电话号码或许会像地址的更改一样经常改变。因此将这些属性放进一个连续时间记录中是很安全的。

现在考虑将职位属性放进记录会发生什么变化。每当一个人更换工作、升职或是调离岗位，或是当公司重组时，其职位就很有可能更改。除非有建立多个连续时间记录的需求，否则职位与其他属性放在一起就不是一个好主意。

图11-9显示了连续的时间段记录的一些特征。

在设计连续时间段记录时要十分小心，因为这时如果属性没有被适当地组合在一起，就有可能导致切实的混乱。下面举一个简单的例子。

图11-10显示了连续时间记录中的一些典型的属性。

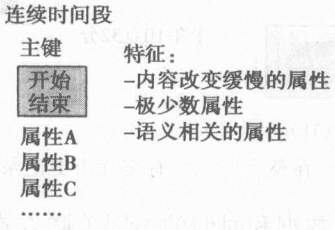


图11-9 连续时间段时间变化数据的一些特征

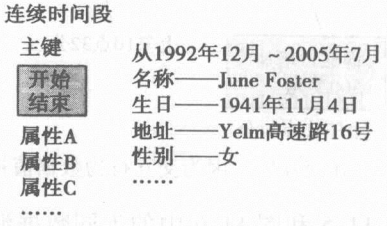


图11-10 连续时间段记录的含义

图11-10显示了姓名、出生日期、地址和性别等属性被放入连续时间段记录。这些数据元素是比较适宜的，因为：

- 在内容上更改缓慢。
- 全部都与个人的描述信息相关。

尽管一个简单的连续时间段记录已经十分有用了，但是还可以将多个连续时间段数据串在一起以在逻辑上形成一个更大的连续记录。图11-11显示了串在一起的连续时间段记录。

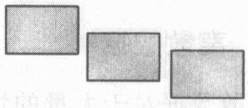


图11-11 将一系列记录组合形成随时间变化的连续记录

11.7 一个记录序列

多个记录形成了一个连续的序列。举例来说，一条记录于2007年1月21日结束，下一条记录开始于2007年1月22日。如此这样，多个记录就从逻辑上形成了一个连续集。

举个简单的例子，2002年7月20日以前June Foster的地址是Yelm高速路。存在一条记录表明了这个地址。然后June搬到了塔斯卡卢萨县B座公寓，她正式的搬迁日期是2002年7月21日，这样就形成了一个新的记录。这两条记录结合在一起就显示了她搬迁的日期和时间以及她所呆过的连续的地址。

尽管可以利用多个连续时间段记录建立一个连续记录，但是并不允许重叠。如果存在记录的重叠，将会导致逻辑上的不一致。例如，如果两条记录都记录了June Foster的地址信息，并且它们有重叠，那么将会说明June Foster某一时段同时居住在两个地方。

11.8 非重叠记录集

图11-12显示了多个连续时间段记录是不允许重叠的。

虽然多个连续时间段记录不允许重叠，但是它们的时间可以是不连续的。例如，1995年时，June Foster去周游世界。那么在这段时间里，她是没有邮寄地址的。她的地址记录只会显示她出发之前和回来之后的地址，但是当她正在周游世界时，她并没有一个固定的地址。

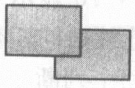


图11-12 不允许有重叠记录

图11-13显示了如果符合数据的实际情况，就允许存在不连续的时间段。

当添加新记录时，新添加的记录属于事务被执行或执行完的那一时刻。依赖于记录的构建方式，调整结束记录有可能也是很有必要的。

图11-14显示了将一条新记录插进连续记录序列的更新过程。

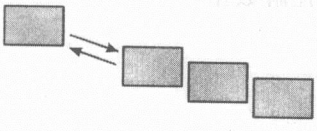


图11-13 未定义的时间间隔是允许的

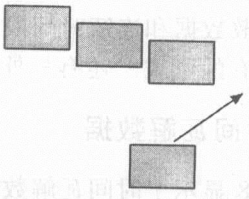


图11-14 当需要更新一组时间段记录时，要添加一个新的记录

11.9 开始和结束一个记录序列

存在多种选择来开始和结束连续时间段记录的序列。

例如，假设最新的记录是从1999年5月开始到现在。然后在2007年4月有一次地址更改，这样就写入一个以2007年4月为开始日期的记录。但是为了保证数据库的同步性，前一个最新记录的结束日期不得不被调整为2007年3月。

总之，一个记录序列可以在任意时间点开始和结束。序列中第一个记录的开始日期可能是一个实际值，也可能是负无穷小。当开始日期是负无穷小时，表明记录包含了从一开始就有的数据。而当序列中第一个记录有一个具体的开始日期时，对任意一个比这个开始日期更早的时间点，都只是不存在数据的定义。

记录序列的结束操作也是和上面一样的方式。一个连续时间段记录的序列中的结束记录的结束日期可能是一个具体值，也可能是正无穷大。当其值被设为正无穷大时，就表明这个记录中的值将会一直被应用，直到再写入新的记录。

例如，假设一份合同的结束日期是正无穷大。这说明这份合同是一直有效的，直到被通知合同结束为止。

图11-15显示了开始和停止一个连续时间段记录序列的一些选项。

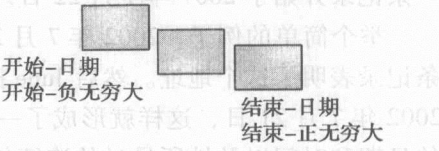


图11-15 对于开始和结束的日期有一些选择

11.10 数据的连续性

离散数据的一个局限就是在数据的两个测量值之间没有连续性。例如纳斯达克周一以2 540点收盘，而周二以2 761点收盘。这样离散的数据就使得不能做出在周二的某个时间纳斯达克会高达2 900点这样的假定。事实上除了每天收盘时的数据外，不能对任何纳斯达克值做任何假定。

图11-16显示了数据的离散性测量缺乏连续性。

而连续时间段数据就没有上述的局限性。借助连续时间段数据，可以对数据的时间连续性作一个判断。

图11-17显示了从连续时间段数据中可以得到数据的连续性。

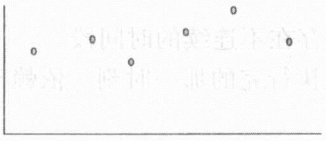


图11-16 离散数据没有连续性信息

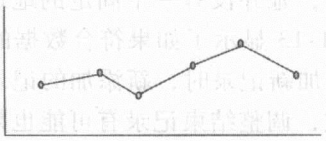


图11-17 连续时间段数据含有连续性信息

虽然离散数据和连续时间段数据是最为广泛的数据形式，但这并不是DW2.0中时间相关数据仅有的形式。还有另外一种数据形式，即时间瓦解数据。

11.11 时间瓦解数据

图11-18显示了时间瓦解数据的一个简单的例子。

在时间瓦解数据中，存在着多种对数据的度量形式。当数据写入系统时，其以小时来度量。然后当一天结束时，会把这一天24小时的值都加起来形成一个一天的值的记录，并将这24个小时的度量值清零；在一周结束时，会建立一个一周的总值并将每天的值清零；在一个月结束时，会建立一个一月的总值并将每周的值清零；在一年结束时，会建立一个一年的总值并将每月的值清零。

在完成这些后，对于小时、日、周等都仅有一个记录集合，因此会节省大量的存储空间。

在基于数据越新，需要的数据细节就越多这样的假设上，时间相关数据的瓦解表现得很好。

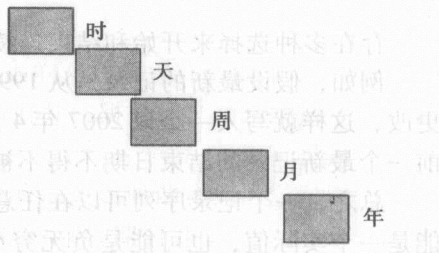


图11-18 时间瓦解数据——另一种时间相关数据形式

换句话说，如果某人要找今天的某个小时的数据，会很容易找到，但是如果要寻找6个月之前的某个小时的数据，就无法找到了。

在很多情况下，这种假设是正确的且数据瓦解很有意义。然而当假设是错误的时候，时间瓦解数据就会产生无法正常工作的环境了。

11.12 归档区中的时间相关变量

DW2.0环境中最后一部分可以应用时间相关的地方就是归档区了。实践中，通常以年为单位存储数据。一年的数据被存储，紧接着又一年的数据被存储。以这种方式分割数据有很多充分的原因。最有说服力的一项原因是数据的语义每年都会有细微的变化。

某年加入一条新的数据元素。第二年会加入一个不同定义的数据元素。第三年又会有一种不同的计算方式。每年总会与之前的年份有细微的变化。

图11-19显示了每一年数据的语义都会有细微的变化。

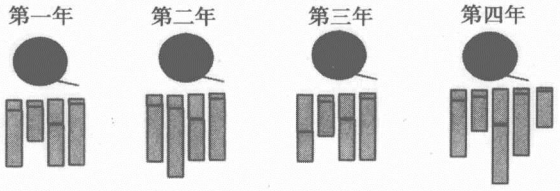


图11-19 归档数据以年为单位存储。注意到每年与其他年份并不是十分地相同

11.13 企业用户的观点

对企业用户来说，数据仓库中的时间相关是很自然和正常的。当一个企业用户想要查询与某个特定时间相关的数据时，终端用户需要提供分析过程的相关时间。

当企业用户想要查询最新数据时，并不输入日期信息，系统就会知道它需要查找最新的数据集。

所以从查询和企业用户交互的角度看，时间相关就像查询本身一样自然和正常。

DW2.0的企业用户得到了比以往所用的环境更强大的分析能力。

DW2.0的结构需要终端用户知道，某一个日期的数据会存在于DW2.0的某一部分或其他部分。例如，对归档区数据和整合区数据的查询，可能会需要不同的查询请求。

然而，企业用户享受到了将旧的、休眠的数据从整合区移除所带来的性能上的好处。因此在DW2.0中，需要在分区数据间进行平衡。

11.14 总结

DW2.0中的所有数据以这样或那样的形式均与某个时刻有关。

交互数据都是当前数据，其在访问时刻是准确的。DW2.0中其他形式的数据记录都带有时间标记。

时间标记一般有两种形式。有的数据附加了一个日期信息，而有的数据则附加了一个开始日期和一个结束日期。第一种数据称为离散数据，而后一种数据称为连续时间段数据。

连续时间段数据可以串在一起形成一个更长的时间段。多个连续时间段记录中定义的

时间段可以是不连续的，但是不允许有重叠。

DW2.0中还存在其他形式的时间相关，例如时间瓦解数据。仅当当前数据需要被访问且在细节上进行分析的时候，时间瓦解数据是非常有用的。随着时间推移，对于细节的需求就会减少。

在DW2.0中还有一种时间相关的形式就是归档数据所用的形式。通常，归档数据按照每年不同的数据定义将数据组织起来。这就允许数据随着时间推移有语义上的细微变化。

第 12 章　DW2.0 的数据流

DW2.0 构架包含许多组成部分，下一代 DW2.0 数据仓库也包含了许多技术。建立一个 DW2.0 数据仓库环境不像盖房子，也不像建立一个小镇，它更像建设一个大都市。

由于 DW2.0 所涉及范围的庞大和复杂性，它很容易使人迷失。人们很容易会仅仅关注并仔细了解 DW2.0 的某一方面。如果这样做了，那么你就会忽略整个构架的更大的“景象”。而有时侯，从构架的细节退后一步去观察更大的“景象”是非常有用的。

12.1　贯穿整个构架的数据流

在整个 DW2.0 构架中贯穿着一个数据流。在许多方面，这个数据流就像人体内的血液流一样重要。数据流满足了 DW2.0 所完成的其他一切事情的需求。

数据流从数据进入交互环境开始。数据可以直接进入交互区，或者也可以通过 ETL 处理进入该区。数据如何进入交互区完全取决于外部应用或 DW2.0 中的应用。

12.2　进入交互区

数据作为面向应用的数据进入交互区。在进入交互区之后，数据被发送到整合区。图 12-1 就显示了进入 DW2.0 环境的基本数据流。

数据流一个令人感兴趣的方面是数据流的速度和传输量。数据流入交互区的速度很快，在外部应用环境中仅仅传输几毫秒就会进入交互区。具有这种性质的输入交易数据可视为实时数据。当然，如果交易是从交互区直接执行，数据就根本没有延时。

在其他情况下，外部应用的交易数据可能需要一个小时或一天才能进入交互环境。进入交互环境时，数据的时间延迟完全由对数据的业务需求所决定。如果有一个合理的业务实例要求数据立即进入交互环境，那么这些数据就应该立即进入。如果没有，那么数据的传送就不必很迅速。

需要重点指出的是：数据进入交互区需要的速度越快，为完成这种快速数据转移的目标所需要的技术也就越复杂和昂贵。

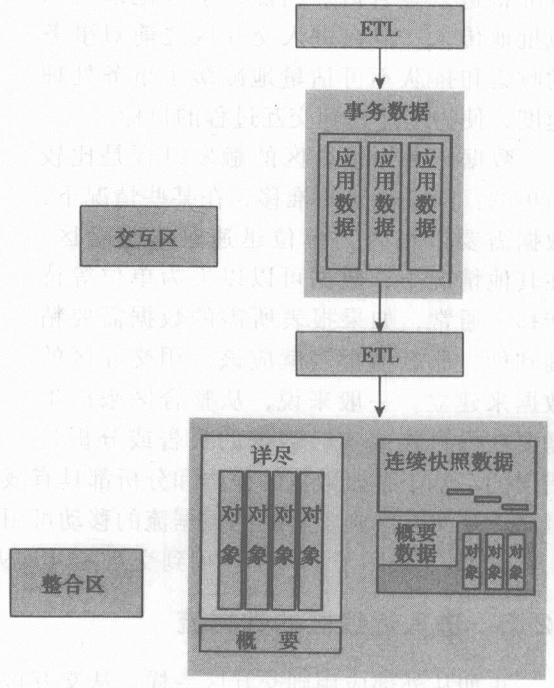


图 12-1　数据传入交互区再从交互区传入整合区

12.3 ETL的角色

来自于外部应用的数据通常由ETL处理进入交互区。当然，数据通过简单的文件传输进入交互区也是可能的，但这不是很常见。数据更可能是经由标准的ETL技术进入交互区。

传到整合区的数据来自于交互区。数据也可能直接传入整合区而不经过交互区。一般的处理都发生在数据从交互区传入整合区的时侯。通常数据是通过ETL处理传入整合区的，ETL处理把数据从一个面向应用的结构重组成为一个企业数据结构。

12.4 进入整合区的数据流

相比于从整合区到交互区的数据流，进入整合区的数据流在速度上更为宽松。数据流以定期的方式（按天，按周，按月，甚至按季）进入整合环境。图12-2描述了进入整合区的数据流的速度。

数据流以小的快速的喷射形式进入交互区，每次就进入一个数据记录。整个文件或大量的记录一次性传入交互区是不常见的。数据传入交互区如同细雨，而不是洪水。

数据进入整合区是由事务的执行开启的。一旦一个事务完成，它所包含的数据就准备进入整合区。当然，事务记录可以成批地传入。但在进入交互区之前对事务的收集和排队不可估量地减缓了事务处理速度，使得不能达到交互过程的目标。

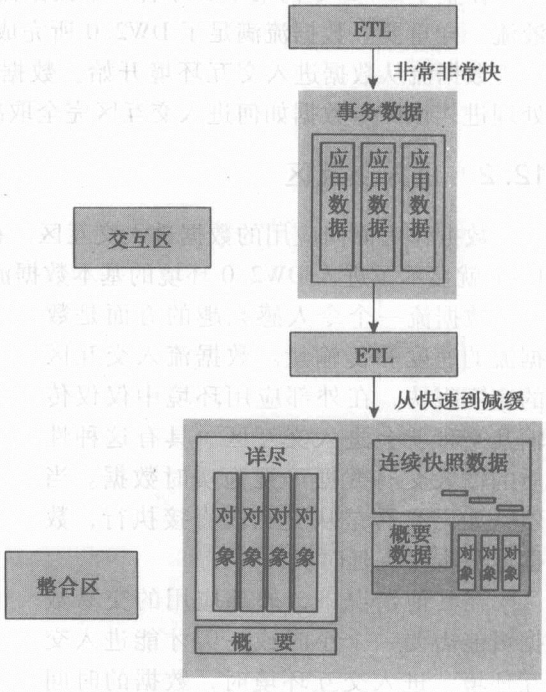


图12-2 传输速度

数据传输到整合区的触发过程是比较简单的：随着时间的推移，在某些情况下，数据需要以小时为单位迅速进入整合区。在其他情况下，数据可以以天为单位等待转移。通常，如果报表所需的数据需要精确到秒，那么该报表就应该使用交互区的数据来建立。一般来说，从整合区来产生需要获得直接或实时数据的报告或分析是错误的。基于整合数据的报告和分析都具有战略性质，不应依赖于精确到秒的实时数据。因此，从交互区到整合区的数据流的移动可用一个较为宽松的安排来完成。

图12-3表示了从外部应用到交互区以及从交互区到整合区的数据移动的触发。

12.5 进入近线区的数据流

正如从外部应用到交互区一样，从交互区到整合区的数据流同样重要。在DW2.0数据仓库中，它们并不是仅有的主要数据流，从整合区到近线区的数据流是DW2.0环境中

另外一个重要的数据流。这种数据流是可选的，并出现在以下几种情况中：

- 整合区有很多数据。
- 交互区中的某些部分数据正在休眠。
- 对整合区数据中有访问的要求。

如果整合区的数据不符合以上的标准，那么就没有必要将其移至近线区。在许多方面，近线区都充当着整合区中数据的一个缓存，当数据不需要很频繁地访问时，就放置在近线区。

近线区基于非磁盘存储。因此，近线数据存储起来不会很昂贵，而且能够容纳大量数据。

图 12-4 表示的是从整合区到近线区的数据流。

来自整合区的数据流一般是比较慢的。通常整合区的数据都会定期地大块移动。例如，每月一次，每次四分之一。

数据访问的概率降低是将整合区的近线区减少了整合区的数据量，这就降低了数据仓库环境的成本并提高了性能。减少整合环境中不被经常访问的数据能够为被经常访问的数据释放整合环境中的磁盘存储。

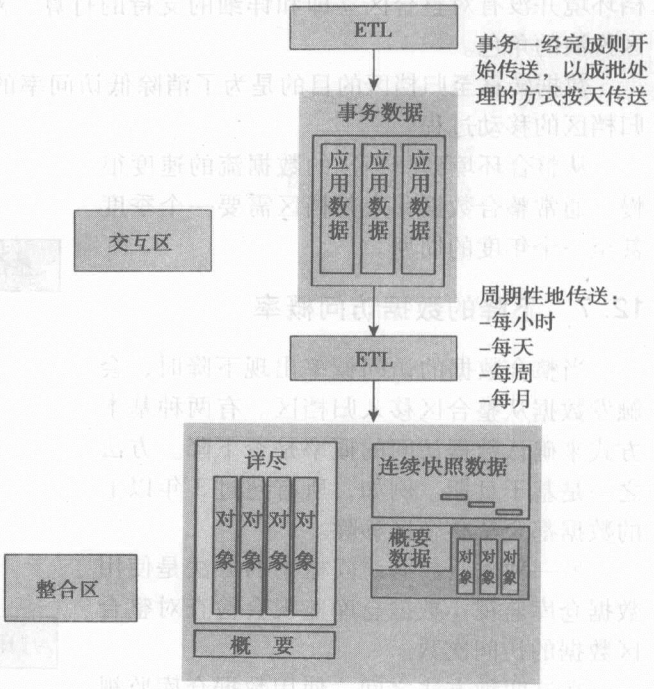


图 12-3 决定数据何时进入不同区域的触发

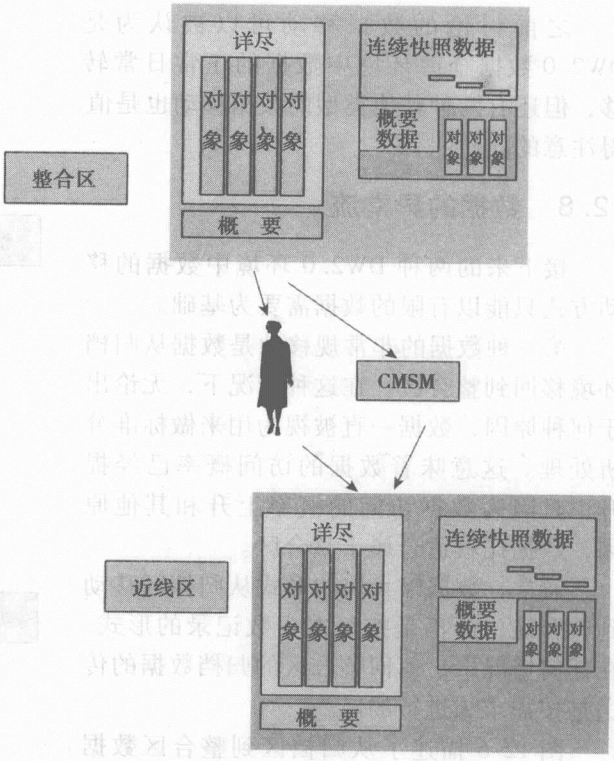


图 12-4 从整合区到近线区的数据流

12.6 进入归档区的数据流

数据同样也可以从整合区传入归档区。从整合区移入归档区的数据和移入近线区的数据有着关键的区别。

当数据移入近线区时，数据结构和数据格式得以保存。这意味着数据在必要时可以迅速和顺利地从近线区回到整合区。近线区的目的在于支持整合区的数据访问。

与此相反，当数据传入归档区，并没有打算让数据迅速地回到整合区。归档区的目的是为了长久地保存数据。在未来的某一时刻，数据

可能需要取自归档环境并保存在某处用于特别分析，甚至可能被送回到整合环境，但归档环境并没有对整合区立即和详细的支持的打算。对整合区的立即和独立的支持是近线区扮演的角色。

数据转移至归档区的目的是为了消除低访问率的数据。图 12-5 表明数据从整合区到归档区的移动过程。

从整合环境到归档区的数据流的速度很慢。通常整合数据移入归档区需要一个季度甚至一个年度的周期。

12.7 下降的数据访问概率

当整合数据的访问概率出现下降时，会触发数据从整合区移入归档区。有两种基本方式来确认数据访问的概率是否下降。方法之一是基于日期。例如，所有超过 3 年以上的数据都要从整合区移除。

另一种确认数据访问概率的方法是使用数据仓库监视。数据仓库监视会检查对整合区数据的访问次数。

在这两种方法之间，使用数据仓库监视是迄今为止最准确的方法。

之前讨论的数据移动可以被认为是 DW2.0 数据仓库环境中数据的正常日常转移，但还有两种其他类型的数据移动也是值得注意的。

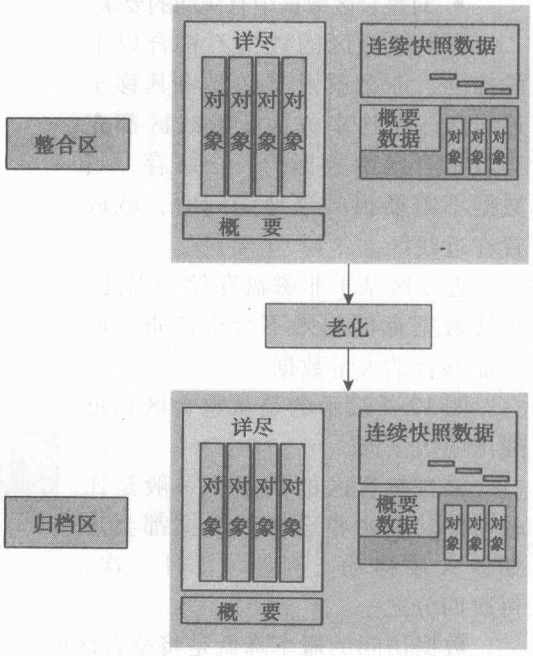


图 12-5 从整合区到归档区的数据流

12.8 数据的异常流

接下来的两种 DW2.0 环境中数据的移动方式只能以有限的数据需要为基础。

第一种数据的非常规移动是数据从归档环境移回到整合区。在这种情况下，无论出于何种原因，数据一直被视为用来做标准分析处理。这意味着数据的访问概率已经提升，且因为数据访问的概率上升和其他原因，数据比较适合属于整合区。

通常，数据以大块的形式从归档区移动到整合区，而不是按每次少数记录的形式。在任何情况下，返回整合区的归档数据的传送是按需求来进行的。

图 12-6 描述了从归档区到整合区数据的移动。

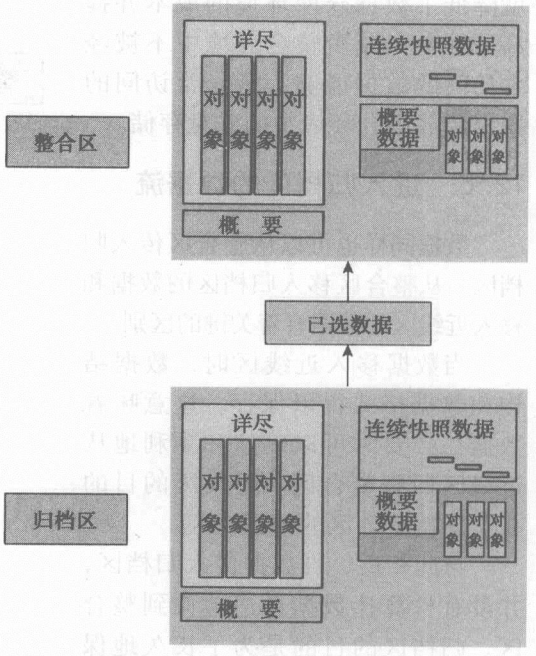


图 12-6 偶尔数据有必要从归档环境移入整合环境

数据的另一种非常规移动发生在近线环境的数据需要返回到整合环境的时候。这种数据传送可以用两种方式来实现。位于这两种环境之间的CMSM软件可以用来管理数据的单个记录的传送。CMSM工具将记录放在整合区中，如果这些记录一直在整合区，那么这些记录就可以显示。这个传送会很快完成，因此不会出现严重的系统性能退化。终端用户可以提交查询，系统能够自动感应到所需的数据有一部分位于近线存储中。然后，系统利用CMSM工具来查找、获取数据，并把数据放在整合存储区，然后执行查询。

从近线区到整合区的全体数据的移动方式还可以按批处理模式来进行。在这种情况下，数据可能被CMSM软件移动或手动移动。在任何情况下，都是因为预期的访问概率上升，旧数据才被送回到整合环境。图12-7描述了从近线区到整合区的数据移动。

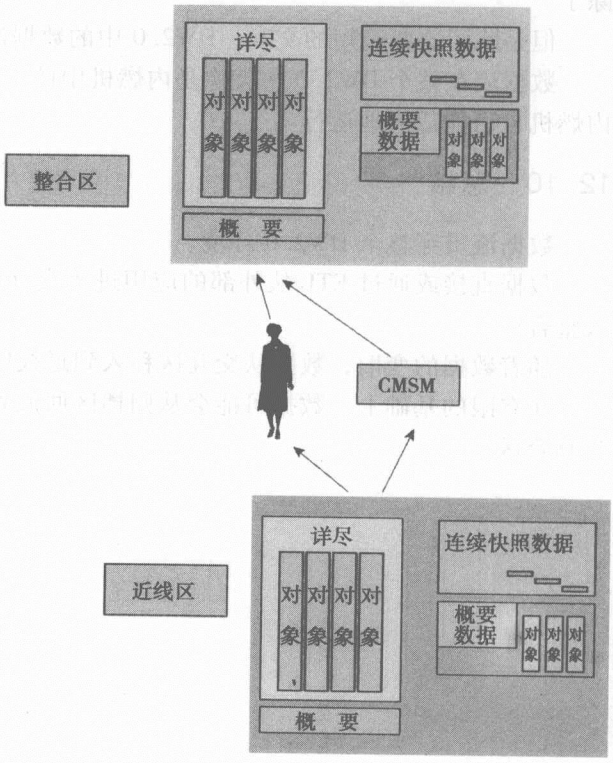


图 12-7 从近线区到整合区的数据移动

在DW2.0环境中，最后还有一种数据的移动值得讨论——从整合区到交互区的数据移动，也可以被称为数据“回流”。

从整合区到交互区的数据移动很少发生。通常参与的数据量并不大。当这种回流发生时，必须做到不能影响在线性能，这是交互环境的一个重要组成部分。图12-8描绘了从整合区到交互区的数据回流。

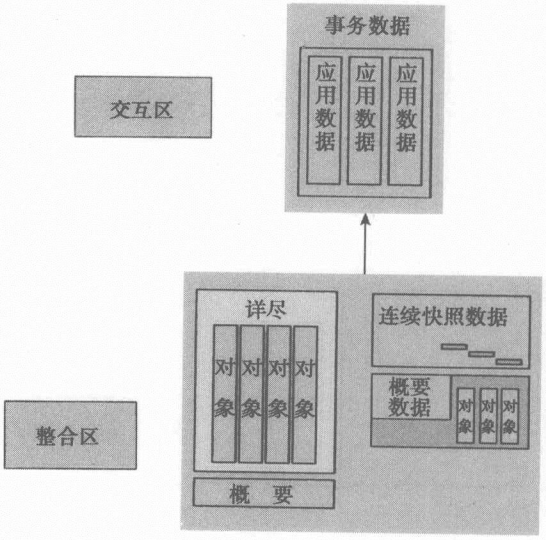


图 12-8 从整合区到交互区的数据回流

12.9 企业用户的观点

数据流是DW2.0环境很自然和正常的特征。企业用户需要意识到有这样一个流动，但他/她不必亲身参与到这些流动机制中去。

只有一种情况，企业用户参与DW2.0中的数据流，就是当企业用户需要提醒系统管理员有必要从归档区或者近线区撤出数据返回到整合区的时候，这种请求被称为配置请求。在提交查询之前，会有一个配置请求。配置请求说明了需要什么样的

数据参数用来从近线区或归档区中选取数据。此外，配置请求还告诉系统数据需要在整合区中保留多久。

有些数据是一项大型研究的一部分，需要在整合区放置很长一段时间，还有一些数据仅需短期保留。一旦数据被放置在整合区并已经被使用，它就可以安全地从整合区移除了。

但是除了这些简短的交互，DW2.0 中的数据流还以未知的速率持续传送至企业用户。

数据流在整个 DW2.0 中就像是内燃机中的汽油流。司机知道有这样一个流，但相信内燃机自己能适当地运行。

12.10 总结

数据流贯穿整个 DW2.0 环境。

数据直接或通过 ETL 从外部的应用进入交互区。来自交互区的数据通过 ETL 处理流入整合区。

随着数据的变旧，数据从交互区移入到近线区或归档区。

在有限的基础上，数据可能会从归档区回流到整合区，并且数据还会偶尔从近线区流入整合区。

第 13 章　ETL 处理与 DW2.0

DW2.0 中最重要的过程之一是 ETL——抽取/转换/装载（extract/transform/load）的处理过程。ETL 处理在数据进入交互区或是数据通过交互区进入整合区时收集、整理并集成数据。ETL 处理过程对于 DW2.0 环境中的日常操作是最基本的过程。

13.1　转换数据状态

ETL 是一种典型的比纯数据流更加强大的东西，是一种改变数据状态的机制。当数据通过并被 ETL 处理转换时，它就经历了一次根本性的状态转变，从应用状态演变到企业状态。这一根本性的改变是 DW2.0 基本理论和存在的核心。

ETL 并不仅仅完成数据的收集和传输工作，其对数据状态的改变也非常重要，这也是 ETL 是 DW2.0 中的基本组件的原因。

13.2　ETL 适用范围

图 13-1 说明了在 DW2.0 中 ETL 处理过程的适用范围。

ETL 处理过程能做很多事，它能从已经多年不被主流 DBMS 使用的技术（如 IMS、VSAM、CICS、IDMS、Model 204 以及 Adabas 等）中收集那些遗留的数据。收集这样的数据不是件容易事，因为每个旧的应用环境的接口都需要采用其自己的技术来完成。

在收集到这些遗留数据后，真正的数据转换工作就开始了。旧的遗留系统的最大问题是，它们在设计之初并不是同构的，在结构、格式、计算、数据定义以及其他语义方面都存在差异。简而言之，要想使旧的应用数据适用于企业数据环境还需要做大量的工作，而正是由 ETL 来完成对旧数据的主要纠正工作的。

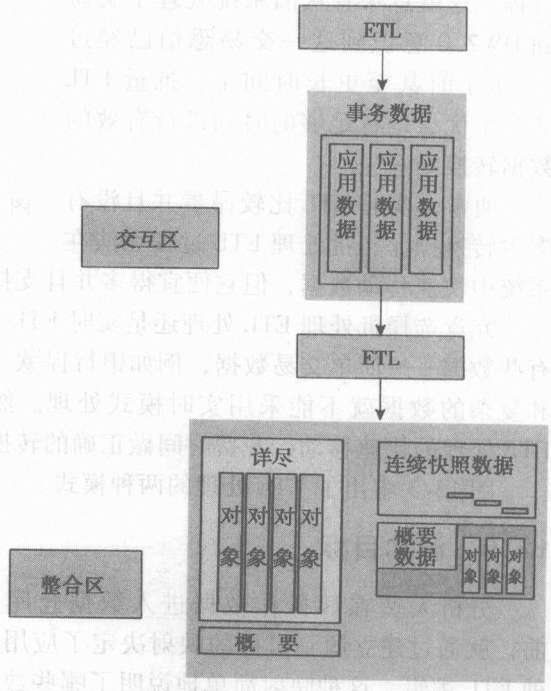


图 13-1　ETL 处理是 DW2.0 中必不可少的一部分

13.3　应用数据到企业数据的转换

图 13-2 描述了 ETL 如何将应用数据转换为企业数据。

ETL 处理能够以两种基本的模式进行操作：在线实时处理和批处理。

13.4 ETL 工作模式

当 ETL 以在线模式工作时，从执行旧数据处理开始，到这一处理反映在 DW2.0 中为止，这段时间以很短的时间单位来度量，比如毫秒。例如，在旧系统环境中，一位银行收纳员在上午 11:32 进行一笔交易，ETL 能同时捕捉到交易已执行这一信息，并且在 10 毫秒内就将该交易移入 DW2.0 环境。该交易被送入 DW2.0 的速度如此之快，以至于其看起来是在旧系统环境中和 DW2.0 中同步发生的。事实上，该交易并没有在执行的同时被 ETL 所处理，只是其发生得如此之快，以至于看起来是同步发生了。

实时 ETL 的问题是，速度通常被看作是主要的成功因素。正是因为速度成为衡量成功的主要标准，所以进入 DW2.0 的数据能够被转换的并不多。

另一种 ETL 处理模式是批处理模式。在该模式下，旧系统中的交易首先被批量存储，然后，在合适的时间（也许是突然）对这一批交易执行 ETL 处理过程。这就意味着从旧系统处理了交易到 DW2.0 意识到这一交易恐怕已经过了 24 小时甚至更长时间了。批量 ETL 的一个优点是有足够的时间进行有效的数据转换处理。

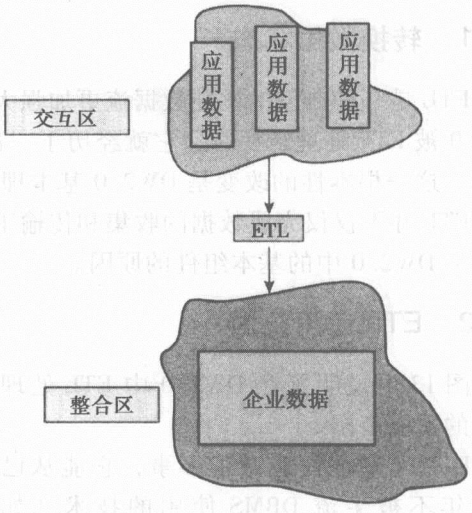


图 13-2 ETL 的实际功能是将应用数据转换为企业数据

通常，实时 ETL 比较昂贵并且没有多少转换操作，批处理 ETL 过程无法在系统中快速移动数据，但它便宜得多并且支持更多的数据转换。

究竟选择批处理 ETL 处理还是实时 ETL 处理，更多的是一种商业选择而非技术选择。有些数据——简单交易数据，例如银行提款——常常采用实时模式处理，而那些更为麻烦和复杂的数据就不能采用实时模式处理。然而，在批处理 ETL 情况下，并不需要考虑 ETL 处理的快速移动，花费时间做正确的转换比快速完成更重要。

图 13-3 给出了 ETL 处理的两种模式。

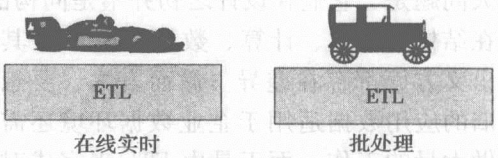


图 13-3 ETL 有两种操作模式

13.5 源和目标

分析人员在源操作数据进入数据仓库之前，就通过建立源到目标的映射决定了应用哪种 ETL 逻辑。这种映射简单地说明了哪些数据必须放置在 DW2.0 中，这些数据来自哪里，对这些数据要做哪些必要的逻辑操作、计算或格式转换。数据源——操作应用系统环境——被称为“源”，数据在 DW2.0 中所处的位置称为“目标”，产生原始数据的源系统称为“记录系统”。

在流向 DW2.0 的各单元数据的源到目标的映射过程中，以下问题是必须要回答的：

- 哪个源系统中的哪个特定单元的数据将构成数据仓库中的数据？

- 如何处理相同数据的多个来源？
- 如果需要数据的默认值必须做些什么？
- 必须做哪些逻辑操作才能使得数据适应企业状态？
- 为将数据重新组织为企业状态下的数据，必须应用哪些计算？
- 为了建立企业数据，必须完成哪些重构或格式转换？

源到目标数据映射的工作需要在 ETL 过程设计开始之前就完成。

13.6 ETL 映射

图 13-4 给出了一个 ETL 数据的源到目标映射。

业务规则是管理映射到 DW2.0 中的数据的必要组件之一，它对送往 DW2.0 的数据有着显著的影响，提示着数据的正确性或质量。但是，不同遗留系统的业务规则不同。因此，ETL 过程必须像裁判一样来决定对于各单元数据而言哪条业务规则凌驾于其他规则之上。

图 13-5 说明业务规则以及源到目标映射对于 ETL 规范来说都是必要的输入。

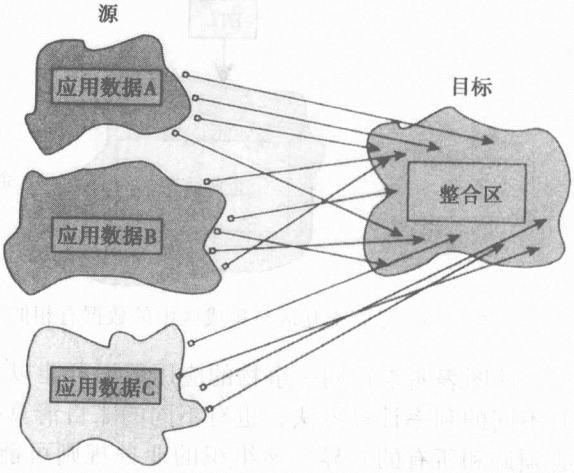


图 13-4 源到目标映射

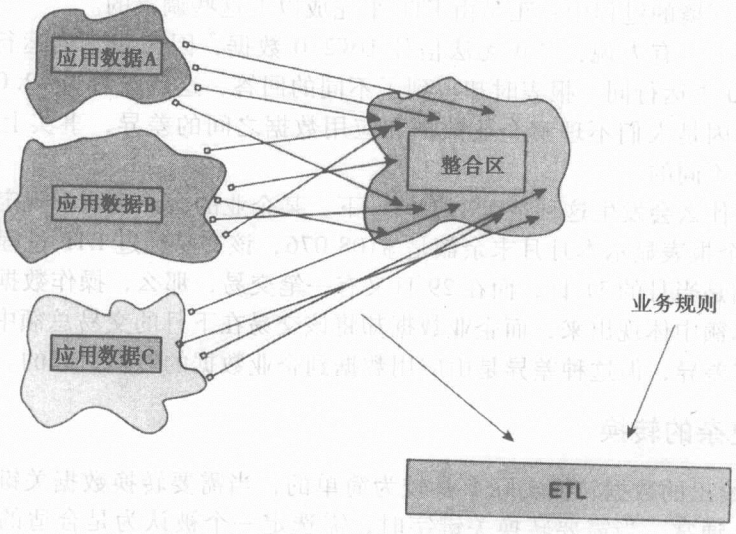


图 13-5 映射与业务规则决定 ETL 处理

13.7 状态转换——实例

数据在经过 ETL 处理过程时转换状态。ETL 处理将数据由应用状态转换为企业状态。

对门外汉而言，这里发生的任何异常情况并不明显。事实上，很多情况下数据只是由一个环境换到另一个环境，因为它的操作状态和企业状态根本没有差别。但是另外一些情况中，需要进行显著的数据转换。

图 13-6 给出了一些简单的转换。

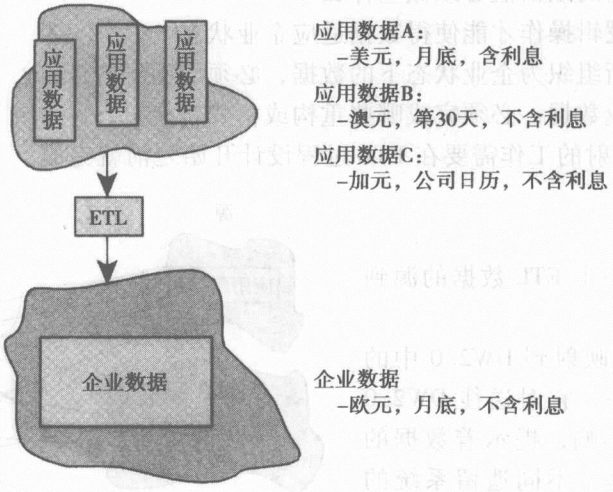


图 13-6 尽管交互区与集成区中的数据有相似之处，但这些数据并非是完全相同或冗余的

该图表明来自同一事物的应用数据可能以不同的形式存在。有不同种类的货币单位，有不同的利率计算方法，也有不同的账目清算截止日期。为了从企业角度了解这些数据，必须调和所有的差异。该组织的业务规则可能是规定在企业数据状态下，现金以欧元来衡量，所有会计工作截止日期都是当月的最后一天，利息不包括在某些计算中。在源数据进入数据仓库环境的过程中，正是由 ETL 来完成以上这些调整的。

在有些情况下，有人说："我无法信任 DW2.0 数据，因为我上周运行了一个报表，然后我在 DW2.0 中运行同一报表时却得到了不同的回答，这意味着 DW2.0 不可信。"产生这种困惑的原因是人们不理解企业数据和应用数据之间的差异，事实上，来自不同源的报表就应该是不同的。

为了理解为什么会发生这种情况，想象一下，某企业的会计系统中规定截止日期是当月的 28 日，一个报表显示本月月末余额是 $108 076，该交易经过 ETL 过程处理。假定企业会计截止日期是当月的 31 日，而在 29 日又有一笔交易，那么，操作数据中会将该交易在当月的交易总额中体现出来，而企业数据却将该交易在下月的交易总额中体现出来。

数据确实有差异，但这种差异是由应用数据到企业数据的转换引起的。

13.8 更加复杂的转换

图 13-6 中给出的数据转换实际上是较为简单的，当需要转换数据关键字时，数据转换就变困难了。通常，当需要转换关键字时，需选定一个被认为是合适的应用程序的关键字，其他应用程序都遵循这种关键字结构。虽然大多数关键字转换都是通过选定一种关键字格式来解决的，但在 ETL 过程中偶尔也需要将数据的关键字完全转换或替换成一种全新的关键字格式。不论以什么样的方法，在设计 ETL 时都必须指定数据关键字转换规则。

13.9 ETL 与吞吐量

数据吞吐量是 ETL 处理另一个关注的问题。即使是只包含执行一个操作的最简单的

ETL 处理，如果吞吐量方面有问题，也可以并行化 ETL 处理。

当一个 ETL 工作流被并行化时，一部分数据经过一个 ETL 处理，另一部分工作流经过同一 ETL 处理的另一个拷贝。ETL 处理和工作流被多次复制，直至获得令人满意的吞吐量。并行 ETL 吞吐流的建立大大削减了整个数据 ETL 处理所耗费的时间。

图 13-7 说明 ETL 处理能够被并行化。

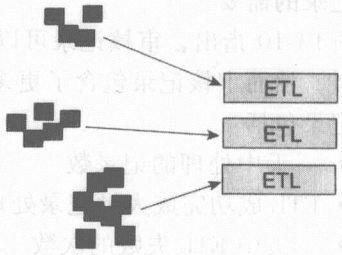


图 13-7 如果吞吐量方面有问题，那么 ETL 可以并行处理以解决这一问题

从理论上来说，ETL 处理必须完成大量的转换活动。其中任意一个转换活动通常都不是很复杂的，但是在数据的一次通行中必须完成所有的转换，对于 ETL 处理而言这的确很复杂。

图 13-8 给出了一些典型的必须由 ETL 处理来完成的转换。

该图给出了许多可能发生在 ETL 处理中的数据转换种类，包括：

- 概括处理
- 日期格式的调整
- 逻辑规范——例如，将“Male/Female”转换成“M/F”
- 数据聚合
- 默认值的提供
- ASCII 转换为 EBCDIC
- DBMS 转换
- 关键字的重构或创建

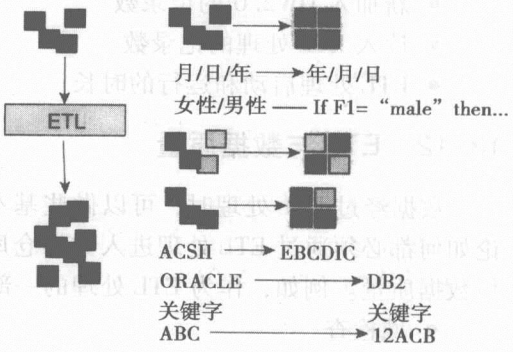


图 13-8 ETL 有主要目的是转换数据

以上仅仅是 ETL 处理需要完成的转换活动中的一小部分。

13.10 ETL 与元数据

尽管 ETL 处理的主要目的是根据应用数据创建企业数据，但它还有一个次要的目的。ETL 还可以用于创建数据转换的一个可审核的记录，即一条被置于元数据中的数据转换审核记录。如图 13-9 所示，元数据成为 ETL 处理一个重要的辅助副产品。

元数据是一种 ETL 处理很容易就产生的副产品。分析人员在设计 ETL 处理时创建的源到目标数据映射本身就是一种元数据的设计。事实上，所有的源到目标数据映射不过就是一个数据转换审核记录，也就是元数据。源到目标数据映射是“关于数据的数据”，是关于数据如何按照自身的方式转换进入数据仓库的具体数据，因此也是一种跟踪 ETL 数据转换的设计。

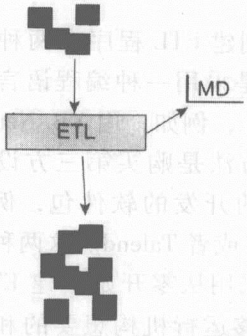


图 13-9 元数据是 ETL 处理的副产品

对于决策支持分析人员而言，ETL 数据转换元数据可能是非常重要的工具。终端用户往往想要从分析用的数据中找到更多信息，元数据作为 ETL 数据转换

处理的副产品，成为满足终端用户需求的容易获得的第一步。

13.11 ETL与审核记录

与在源数据经过ETL处理时获取和提供可用的元数据同样重要的是，对ETL处理的审核记录的需要。

图13-10指出，审核记录可以在元数据经过ETL处理后留下来。审核记录与元数据非常相似，然而审核记录包含了更多关于数据经过ETL处理的详尽数据。典型的数据转换审核记录包括：

- 一天中处理的记录数
- ETL成功完成大批记录处理工作的指示
- 一天中ETL失败的次数
- 任何ETL失败的原因
- 如何解决失败的描述
- 新加入DW2.0的记录数
- 读入ETL处理的记录数
- ETL处理启动和运行的时长

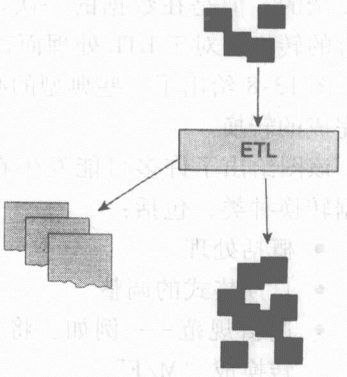


图13-10 ETL处理总会遗留一些审核记录

13.12 ETL与数据质量

数据经过ETL处理时，可以做些基本的数据质量检查甚至编辑工作。由于源数据无论如何都必须通过ETL处理进入数据仓库，所以在ETL中只需要少数额外的资源用以评估数据质量。例如，作为ETL处理的一部分，以下的数据质量检查就很容易做到：

- 域检查
- 范围检查
- 合理性检查

应当指出的是，ETL处理中做的任何数据质量保证都仅限于对单个数据记录的处理中。换句话说，如果需要多条数据记录来确定数据合法性或质量，那么这种数据质量的核查/检查对ETL来说就难以做到了。

图13-11说明数据质量检查可以作为ETL处理的一部分来完成。

13.13 创建ETL

创建ETL程序有两种基本的方法：一种方法是采用一种编程语言或工具从零开始创建ETL，例如，用VB.Net、C或者C++；另一种方法是购买第三方设计的旨在用于ETL处理的开发的软件包，例如Ascential、Informatica或者Talend。这两种方法都各有利弊。

采用从零开始创建ETL处理的优点在于它能够运行机构想要的和具体规定的任何逻辑操作，任何能被编程实现的东西都可以包

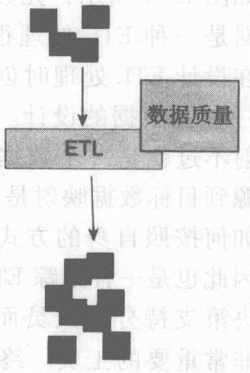


图13-11 ETL为数据质量检查提供绝佳的机会

含在一个 ETL 中。它的缺点在于绝大部分 ETL 程序都是标准例程，不需要作任何规定。此外，随着时间的推移，维护代码是一种不佳的、昂贵的、消耗资源的方法。

采用第三方软件的优点在于能够比上述方法更迅速地建立 ETL 处理。另外，第三方软件的代码更容易维护。问题在于几乎总需要那么一些专门的数据转换算法，它们是如此复杂，以至于难以整合进任何之前已经编程实现的第三方技术中。

13.14 代码创建或参数驱动的 ETL

一些 ETL 在代码创建的基础上完成，另一些则在参数驱动程序的基础上完成而不产生专用代码。图 13-12 给出了实现 ETL 的不同选择。

13.15 ETL 与丢弃

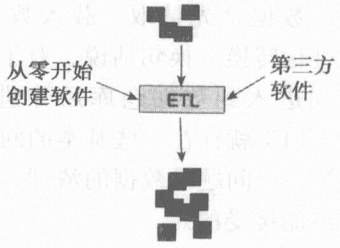


图 13-12 ETL 可使用从零开始创建软件或第三方软件或两者的结合

对于 ETL 而言，如何处理丢弃的数据是其面对的主要挑战之一。当源系统记录包含不正确的值时会发生什么？处理这种情况有许多方法，每种方法都各有利弊。

第一种方法是根本不允许丢弃的数据进入数据仓库。尽管这种方法维护了数据仓库中的原始数据，但它同时也将那些可能正确且有用的数据阻挡在外。例如，假定一个有 10 个要素的交易，其中一个要素是不正确的，那么是否值得将其他 9 个正确的要素从数据仓库中排除出去？

第二种方法是为那些已知不正确的数据提供默认值，这种方法允许所有的其他附随的有效数据进入数据仓库。一个重要的警告是，这种方法可能会在做数据汇总或其他数据分析时产生非常不规则或不可靠的结果。

第三种方法是将无效数据放入数据仓库并做标记。这样分析员就会知道该数据是无效的。

第四种方法是建立一个丢弃文件。只要丢弃文件能够被自动整理，这种方法就不失为一个好方法，但是如果丢弃记录需要手动来整理，那么这种方法就不好了，不建议使用。

图 13-13 描述了丢弃文件的建立过程。

13.16 变化数据的捕获

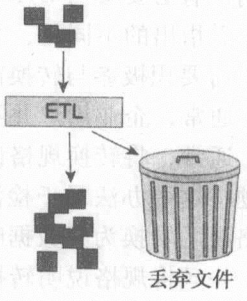


图 13-13 丢弃文件需谨慎处理

变化数据的捕获（changed data capture，CDC）是输入到 DW2.0 交互区中的一种特殊数据，由产生交易而创建了日志磁带时得到 CDC。从日志磁带中捕捉交易活动比在文件中寻找可能进行的活动要方便得多，也准确得多。

但是为了这样的目的去使用日志磁带存在一些问题。第一个问题是操作往往不愿意放弃那些等于是它们必要的基础技术的东西。在备份和恢复过程中需要用到日志磁带，如果操作舍弃这些磁带，将会有灾难性的后果。

第二个问题是创建备份日志文件用于数据恢复，而不是为了向数据仓库中输入数据。

日志磁带的格式最好是保密的，并应非常难于破解。

图 13-14 给出一个作为 ETL 处理输入的日志磁带的用法。

13.17 ELT

抽取/装载/转换（extract/load/transform，ELT）处理与 ETL 的联系相当紧密，却又不完全相同。ETL 和 ELT 之间的差别在于：在 ETL 中数据经过抽取、转换后再载入数据仓库中；而在 ELT 处理中，数据是先抽取、载入数据仓库中后再进行转换。换句话说，对于 ELT，数据直到进入到数据仓库后才进行转换。这样，ELT 就存在一些基本的问题。

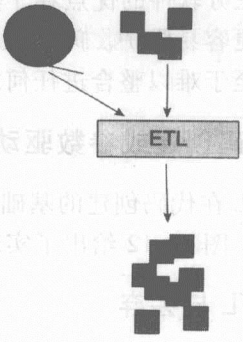


图 13-14 对于变化数据的捕获来说日志有时很有用

第一个问题是数据的清理。完全有可能在抽取并装载数据之后忘记做转换工作，这当然是不能接受的。

在数据被载入数据仓库后做数据转换时，ELT 方法还会引起数据完整性问题。在某时刻，某些数据单元的值为 100，而在另一时刻为 35，这样数据仓库就失去其可信性了。也就是说，使用 ELT 的数据仓库中不存在数据完整性。

图 13-15 给出了一个 ELT 处理。

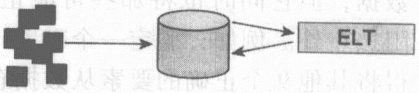


图 13-15 ETL 的另一形式是 ELT

正是由于自身固有的问题，ELT 处理并不是一个好的构架选择。

13.18 企业用户的观点

ETL 是 DW2.0 的一部分，企业用户不参与其中。在 DW2.0 的其他大部分方面，企业用户偶尔会来观察。在大多数其他情况下他/她知道系统内部发生了什么，但却不积极参与。

但对 ETL 并不这样，由于数据由应用状态转换为企业状态要经过 ETL 处理，所以企业用户有必要参与 ETL 处理。很多情况下，只有企业用户才知道如何正确地进行转换。

需指出的不同是，企业用户拥有最终的批准权来确定如何进行转换工作，因此，企业用户需要积极参与转换需求的创建。

通常，企业用户并不实际执行 ETL 过程，而是告诉技术员应当如何进行转换。

通常，将转换规格说明写下来是个不错的主意，这样做是很重要的。万一转换后发生问题，这一办法对于检测错误发生源很有效。该方法在另一种情况下也很重要，即在将规格说明转换为元数据时。

一旦将规格说明转换转变成元数据，它们将被提供给那些想要看到遗留数据的分析人员。“这个数据从哪里来，是什么意思？”有了能够反映转换过程的元数据，回答这样的问题就变得很容易了。

因为企业用户了解数据，所以他们被视为转换方面的权威，常称其为数据管理员。理想情况下，DW2.0 环境中的每个数据单元都有且仅有一个数据管理员。换句话说，每个

数据单元都有一个不同的数据管理员。

数据管理过程是一个持续的过程，虽然 DW2.0 环境开始时就拥有数据管理员是很重要的，但应当认识到，在整个 DW2.0 生命周期中数据管理员也是不可或缺的。

数据管理员不负责数据库的日常维护和管理。如果数据库装载失败，应该由系统管理员来完成修复错误的工作。相反，数据管理员负责数据表中数据的日常维护和管理。例如，如果一个人的年龄被列为 1000 岁，那么除非他的名字是 Methuselah（《圣经·创世记》中的人物，据传享年 965 岁），否则数据管理员就负责修正这一错误，并确定该错误是如何进入数据库的。

13.19 总结

ETL 是用于转换数据状态的过程。数据通过 ETL 处理后由应用状态转换为企业状态。ETL 处理发生在数据进入交互区时，在数据由交互区向整合区过渡时会再次发生。

ETL 可以以在线模式或批处理模式运行。以在线模式运行时强调的是数据的移动，而以批处理模式运行时强调的是数据的转换。

数据的产生地被称作源，数据的去处称为目标，显示数据如何从源到目标的逻辑称为映射，全部数据的源集合称为记录系统。

在大量数据需要进入目的地时，ETL 处理可以以一种并行的方式运行。

ETL 的一个副产品是用于描述数据移动的元数据，它的另一个副产品是审核记录。

除了数据转换之外，ETL 处理还包含了简单的数据质量检查。

ETL 处理可以完全自主开发实现，也可利用第三方厂商提供的软件来实现。

有时，在处理由多项交易过程组成的源时，日志文件可作为 ETL 处理的输入。此时，日志文件常被称作变化数据的捕获（CDC）。

第 14 章　DW2.0 与粒度管理器

几乎所有来自外部源的数据都是通过 ETL 处理传递到交互区的，虽然偶尔也直接传递到整合区。但是一些少见的情况下，我们也需要通过另一种方式来传递来自外部环境的数据。

14.1　粒度管理器

这种处理机制就叫做粒度管理器。粒度管理器所做的工作与 ETL 处理截然不同。

为了了解 ETL 处理是如何工作的，图 14-1 在概念层次上描述了一个普通的 ETL 处理。

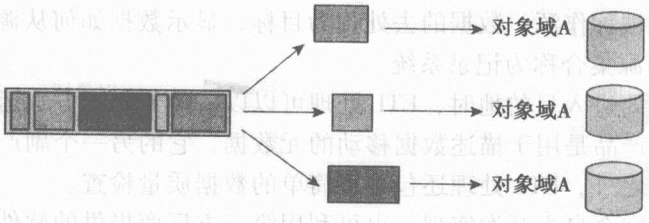


图 14-1　数据的正常处理过程

图 14-1 代表在普通 ETL 处理中对象域对不同种类数据的习惯性存储。ETL 读入一个源记录，然后将该记录分成几个不同部分，根据它们所依赖的对象域的不同，分别将这几部分发送到不同的目的地。例如，假设存在一笔交易，它产生了一条关于销售某件产品的记录。该产品的销售记录作为一条单独的记录到达 ETL 处理，然而这条记录却是各种不同数据所组成的信息集合，例如收入信息、产品信息以及顾客信息等。ETL 处理知道需要把这些不同类型的信息发送到不同的目的地。于是产品交易信息被发送到公司，而面向对象的数据则作为源数据被载入到目的数据仓库中。这是 ETL 处理最基本也是最重要的功能之一。

14.2　提高粒度级别

数据粒度管理与 ETL 处理是截然不同的。与将数据从单一的记录拆分成多条然后送入面向对象的数据单元不同，粒度管理器实际上是一个合并数据的过程。合并或统一数据的需求并不是经常出现。但是在某些罕见情况下，当外部世界的源数据细节程度太低时，这些数据就必须被统一。在使用因特网时产生的点击流数据就是一个需要使用粒度管理器的绝好例子。

当网站处理行为被追踪时，鼠标的每次移动、页面的每次跳转以及每次进入一个新的链接都会产生一条点击流记录。这些发生在网站被追踪时的活动数据的细节程度被降到最低。这些低级别的点击流数据往往存在很大的问题，绝大部分数据都没有任何商业价值。

据估计，最终90%的点击流数据都是无用的。点击流数据追踪的特点就是产生以及获得了大量无用数据。这些在点击流数据中存在的无用数据是一种极大的没有任何必要的开销。

另一个常见的由于粒度太低而需要使用粒度管理器的例子是模拟计算机收集手工数据。大部分模拟数据都是可有可无的，但是也有小部分是极其重要的。这些重要数据的粒度非常低，但是又必须在DW2.0环境下使用，因此它必须在使用粒度管理器处理之后才能被导入。

14.3 过滤数据

图14-2说明大量的数据通过一个类似点击流的处理机制进入工作流，然后被“过滤”、聚集或者合并。

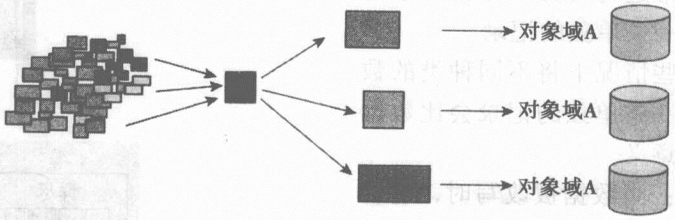


图14-2 有些数据对于数据仓库来说粒度太低

当在DW2.0环境下有多个地方需要使用时，就可以放置粒度管理器。图14-3说明当数据进入交互区时粒度管理器可能被放置的位置之一。

粒度管理器可能被放置的位置之二是数据直接进入整合区的位置。这种情况发生在数据不需要在进入整合区之前在交互区做停留。图14-4描述了当数据直接进入整合区时粒度管理器的放置位置。

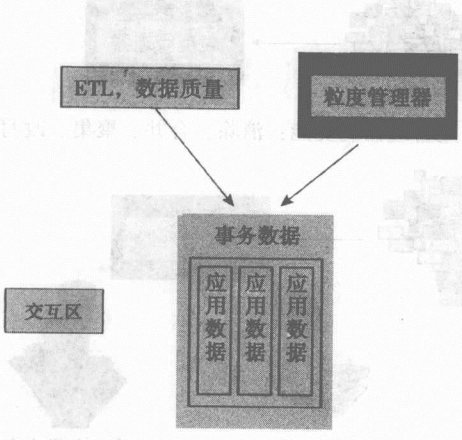


图14-3 粒度管理器放置的位置之一

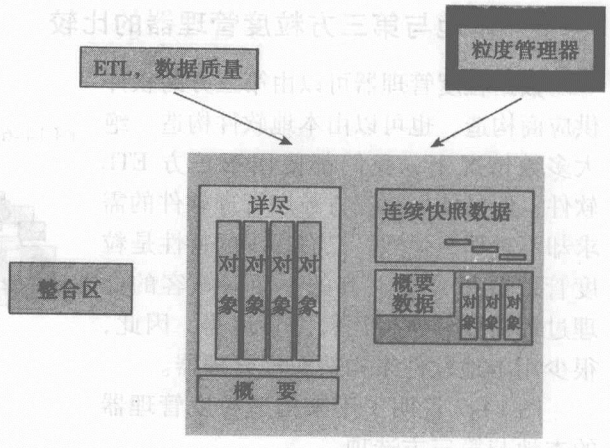


图14-4 粒度管理器放置的位置之二

还有一个位置粒度管理器偶尔会使用，那就是数据被传递到归档环境时的位置。图14-5描述了当数据进入归档环境时粒度管理器所担任的角色。

当数据被传递到归档环境时使用粒度管理器是一种非常少见的情况。只有在整合区发

生大规模的交易，并且这些交易的细节永远不需要用于分析的情况下，使用粒度管理器才有意义。

14.4 粒度管理器的功能

粒度管理器实现的功能是非常直观的。它至少能实现以下功能：

- 消除不必要的数据：那些在未来对公司没有任何参考价值的输入记录将被丢弃，这些数据预计占了总输入数据的90%。
- 合并：那些对公司有参考价值的数据可以经常进行合并，即将多条记录合并为一条单独的记录。
- 聚集：某些情况下将不同种类的数据聚集成一条单独的记录会比数据合并更有意义。
- 改写数据：当数据被改写时，它会以一种格式和结构输入而以另一种格式和结构输出。改写那些原本粒度很低的数据是一件非常常见的事。

这些都是粒度管理器所完成的动作。这些动作的直接结果就是极大地压缩了数据并且剔除了无用数据。图14-6说明了粒度管理器的功能。

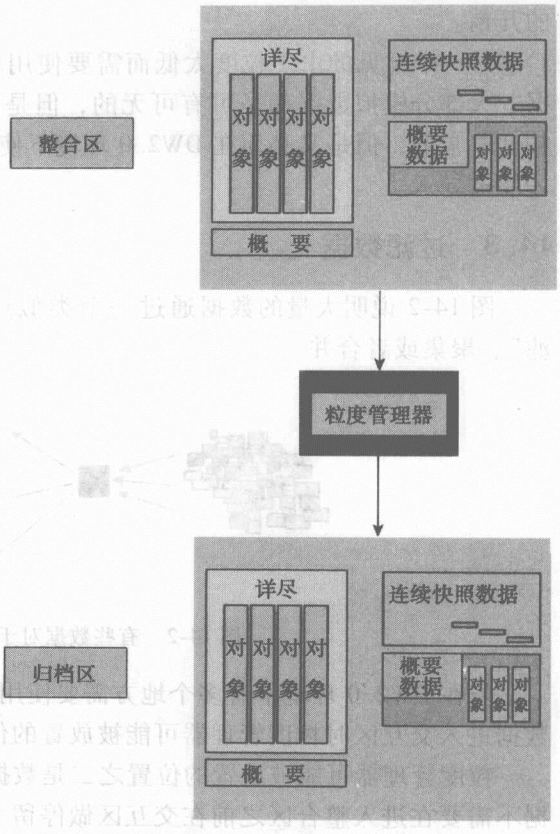


图14-5 粒度管理器放置的位置之三

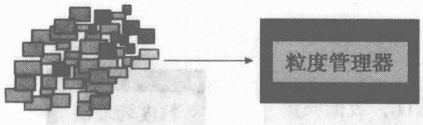


图14-6 粒度管理器的功能：消除、合并、聚集、改写

14.5 本地与第三方粒度管理器的比较

数据粒度管理器可以由第三方的软件供应商构造，也可以由本地软件构造。绝大多数情况下，我们都使用第三方ETL软件，但是使用第三方粒度管理软件的需求却不是那么强烈。只有很少的特性是粒度管理器和ETL共有的。面向顾客的处理过程需要经常使用粒度管理器，因此，很少用本地软件来构造粒度管理器。

图14-7说明了用来构造粒度管理器的本地与第三方选项。

图14-7 构造粒度管理器的两种基本方法

14.6 粒度管理器的并行化

有时会有大量的数据需要通过数据粒度管理器来处理，那么就有可能需要通过并行地运行粒度管理软件来减轻处理负担。通过并行地运行两个或者更多的数据粒度管理软件

实例，处理数据所需的时间将会大幅度减少。

图 14-8 描述了并行运行中的数据粒度管理器。

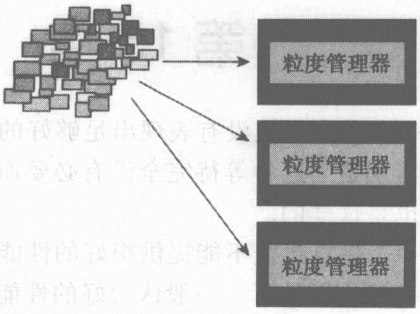


图 14-8 如果值得可以并行地运行粒度管理器

14.7 作为副产品的元数据

除了可以将数据压缩到一个合适并且有效的大小外，粒度管理还能够被用来生成元数据。图 14-9 描述了在粒度管理中作为副产品生成的元数据。

在粒度管理中作为副产品生成的元数据可能包含以下信息：

- 哪些数据被丢弃了。
- 哪些数据被合并了，合并后的记录包含哪些内容。
- 哪些数据被聚集了，聚集后的记录包含哪些内容。
- 数据是如何被改写的，改写后的记录包含哪些内容。

元数据汇总了粒度管理的处理结果。

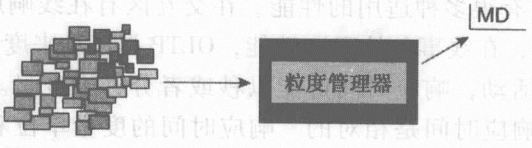


图 14-9 在粒度管理中作为副产品生成元数据是非常有效的

14.8 企业用户眼中的粒度管理器

对于企业用户来说，粒度管理器与终端用户之间的联系并不大。唯一的联系存在于粒度管理器生成的转换规范。

终端用户在转换规范这个任务上与粒度管理器承担相同的责任，而企业用户则负责 ETL 规范。

14.9 总结

有时外部的数据对于 DW2.0 环境来说粒度太低，这时就需要在数据被载入数据仓库之前对它使用粒度管理器，从而提高数据的粒度级。

当数据进入 DW2.0 环境或者在 DW2.0 的各区之间传递时，粒度管理器就负责对数据进行过滤、合并、聚集或者重构。

数据粒度管理器能够并行地运行。它可以由本地软件构造，也可以通过第三方软件供应商构造。

元数据是粒度管理过程中产生的副产品。

第 15 章　DW2.0 和性能

一个信息系统没有表现出足够好的性能是一件令人很厌恶的事情。大部分人都讨厌等待，特别是当这种等待完全没有必要而且毫无收益时。在 DW2.0 环境下人们对待等待的态度也是这样的。

一个信息系统不能提供很好的性能就可以认为它是低效的。如果性能十分不好，那它就是毫无作用的了。一般认为好的性能表现是好的信息系统的一个特点。

但是好的性能并不像一个新功能那样可以被很容易地添加到系统中。好的性能包含了许多方面，应该从整体上进行设计并且从一开始就加入系统中。

15.1　好的性能——DW2.0 的基石

好的性能对于一个有效的 DW2.0 数据仓库是至关重要的，并且在整个 DW2.0 环境中都很有必要。

对于 DW2.0 来说，有很多种适用的性能。在交互区有在线响应性能或者 OLTP（on-line transaction processing，在线事务处理）性能，OLTP 是以秒来度量的。在整合区有分析性能。对于不同的分析活动，响应时间可能以秒或者分钟来度量。而在归档区，响应时间则是以天来度量的。响应时间是相对的。响应时间的度量单位和期望值随着任务的内容和地点的不同而不同。但是在整个 DW2.0 环境下响应时间都是很重要的。

15.2　在线响应时间

在线响应时间正常的期望值是 2 ~ 3 秒。在线响应时间又经常被称为实时响应时间，它是在交互区中完成的。

图 15-1 描绘了在线响应时间。

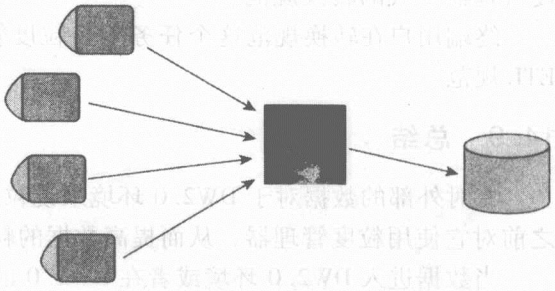


图 15-1　事务响应时间——2 ~ 3 秒

在线响应时间是这样衡量的：从用户按下向计算机发送事务的键开始到返回用户的第一个响应之间的时间。从事务的键入到第一个响应的返回之间发生了许多处理过程。事务被传输到网络上，然后网络将事务传输至系统。在通过安全协议后，事务进入到一个队列中。这个事务会在队列中等待，直到执行该事务所需的系统资源都可用。事务进入计算机中用于处理事务的应用部分。计算机中的代码被执行，对于数据库的调用也通过 DBMS 完成了。数据库收集数据，这些数据被返回到计算机的应用部分。计算机的应用部分运用事务执行逻辑操作，做出决策，并可能向数据库中写更多的数据。最后，事务被执行完成，离开应用部分，最终的结果则通过网络被传回最初发起事务的用户。为了达到好的响应时间方面的性能表现，所有这些活动必须在几秒之内

完成。

影响事务性能的因素包括：

- 冗长的活动，比如网络传输。
- 数据库的输入和输出。
- 在队列或其他地方的等待。
- 事务本身的大的任务量。

15.3 分析响应时间

DW2.0环境的另一个性能考虑被称为分析性能的度量。分析性能问题涉及整合区、归档区，偶尔也和近线区相关。

分析响应时间的期望值一般在10秒至1小时之间。假设对分析型处理给出了大量时间用于性能度量，则通常认为分析环境下的性能是很容易实现的。但根本就不是这么回事。

分析环境和事务处理环境是非常不同的。在分析环境中：

- 做出的决策的战略性要强于战术性。
- 分析型处理所需的数据量要远多于事务型处理所需的数据量。
- 在整天中可预知的数据流动相对更少。

存在于分析环境和事务处理环境之间的这些根本的区别，使得可以采用更加随意的方法进行分析响应时间的性能度量。

15.4 数据的流动

整个系统中的数据流大体和事务处理中所描述的相同。这两种数据流关键的不同在于以下两个地方：

- 分析型处理所需的数据量要显著多于事务型处理所需的数据量。收集大量的数据用于分析需要大量的数据输入/输出操作，而这样的输入/输出操作会明显减慢计算机的速度，因为输入/输出操作是机械级的速度，而不像其他计算机内部操作那样是电子级的速度。
- 分析型事务的排队等待时间取决于在队列中等待被执行的其他分析活动。对于分析活动，在队列中的时间是不可预知的，并且可能很长。

因此，对于分析活动，响应时间一般比较长并且不可预知。

图15-2就说明了针对数据的正在进行的分析活动。

有关性能的任何讨论都需要说明差的性能的影响。对于事务型处理和分析型处理，差的性能所导致的影响是非常不同的。

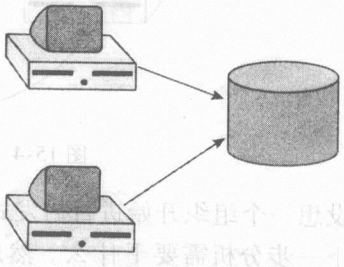


图15-2 分析响应时间——15秒到24小时

15.5 队列

当事务型处理的性能变差时，将会在公司与客户交互的地方对公司的业务产生影响，如队列等待。设想一家银行柜台或航空公司登机口，在

正常操作的情况下，当顾客到达银行或机场时，只有很少人等在他的前面。在情况好的时候，也许只有两三个人在他的前面，但在情况不好的时候，可能会有20多个人在他的前面。这样来看，顾客在队列中向前移动以及接受服务的速度就是等待队列的长度以及他所在位置的函数。

为了实现平稳运行，银行的营业员以及航空公司的服务人员在处理顾客业务的时候必须有令人满意的响应时间。响应时间必须是及时的、一致的。当处理响应时间比较短时，营业员可以在较短时间内服务每一位顾客，顾客等待队列也会以一个稳定合理的速率前进。

但是还要考虑在银行的营业员以及航空公司的服务人员并没有达到很好的响应时间的情况下，将会发生什么。接待每一位客户都需要很长的时间，这对于正在接受服务人员服务的顾客来说是比较糟糕的，对于处于等待队列中的顾客来说更加糟糕。队列会不断加长，直到人数多得不能接受。

因此真正对事务响应时间产生消极影响的不是已给事务的执行，而是等待队列中等待执行的事务的积累效应。事务处理性能较差以及长时间的队列等待时间所带来的最坏的问题就是这种影响已经被公司的用户直接感知到。

图15-3说明了当基本的事务型处理出现问题时，会形成一个令人无法忍受的长队列。

在分析型环境中，低下的性能表现同样也会产生消极影响，它会影响分析者为管理层准备信息的效率。

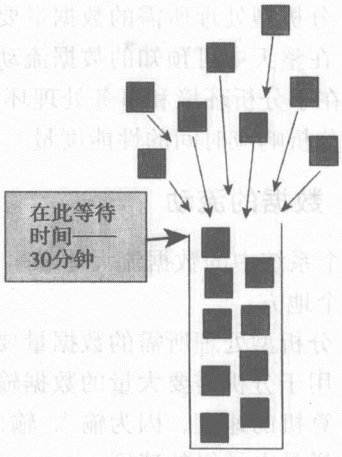


图15-3 当在线事务响应时间表现较差时将会对业务产生消极的影响

15.6 启发式处理

较差的性能表现对于分析团体的消极影响和分析型处理的实现方式有关。图15-4描述了分析型处理的启发式特性。

当启发式地完成处理时，对于分析活动就几乎没有什么计划安排。这是因为，在启发式分析中每个步骤都需要完全依靠上一步骤的完成效果。

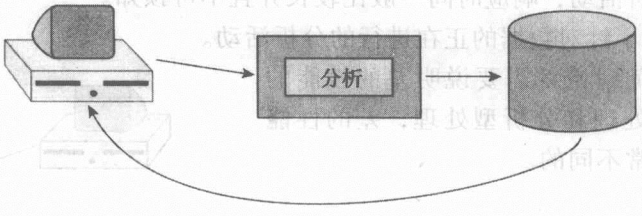


图15-4 分析型处理的精髓就是启发式处理

设想一个组织开始进行启发式分析。得到第一个步骤的结果并加以分析，只有这时才清楚下一步分析需要干什么。然后完成分析的第二个步骤并得到更多的结果，这时分析两步的结果，第三步的分析所需要的活动也就清晰了。这个过程一直持续直至达到最终

的结果。

事实上，在启发式处理时，分析过程中是没有组织、没有计划路径的。

15.7 分析的生产率和响应时间

启发式分析完成的速率完全取决于分析型处理完成的快慢。也就是说，分析型处理完成得越快，就能越快地得到最终结果。

图 15-5 对比了两个分析者的性能表现。

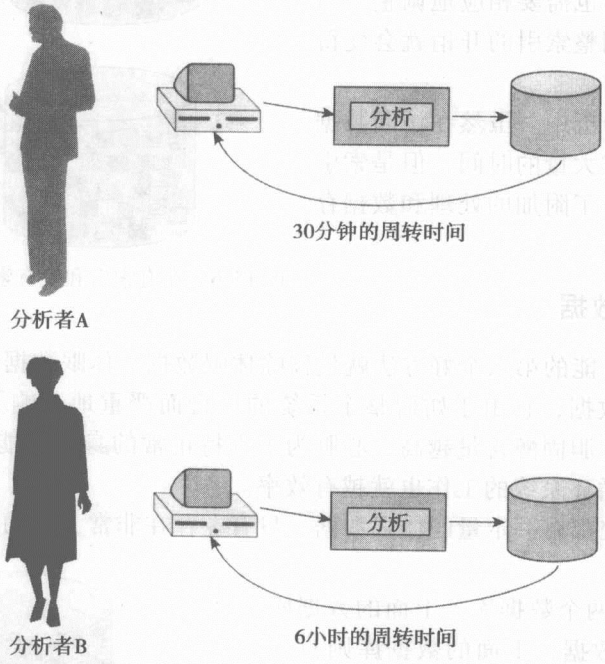


图 15-5 考虑两个分析者只在分析周转时间上不同的生产率

在图 15-5 中，两个分析者唯一的不同就是其分析活动完成的速率。分析者 A 可以在 30 分钟内完成一个启发式分析，而分析者 B 则需要 6 个小时完成一个启发式分析。

事务型处理和分析型处理在差的性能表现方面还有一个大的不同。在事务型处理中，差的结果可以被公司的顾客直接感受到。而差的分析型处理效果是在公司内部被分析人员感受到。对于公司而言，差的事务型响应时间在短期看来更具有破坏性。

可以从很多方面入手来达到好的性能表现，因此对其需要一个整体的方法。性能必须同时在多个方面提升。下面就介绍在 DW2.0 环境下为了达到长期的良好性能需要做的一些工作。

15.8 索引

针对数据仓库性能的一个最简单的设计方法就是创建数据索引。如果创建了索引，系统就不用为了定位和获取所需信息而对所有数据进行顺序查找。

图 15-6 描绘了数据的索引。

图 15-6 的上部显示了一个没有索引的数据库。当一个人访问该数据库时，为了查找

某个记录他要搜索整个数据库。

图15-6的下部显示了一个建立了索引的数据库。当一个人在数据库中查询某些数据记录时，会首先查阅索引。如果在索引中可以直接找到对应的数据信息，就可以直接访问它，这样就极大地减少了数据访问时间。

似乎为数据库中的所有数据都建立索引是一件好事。但是，索引数据也是有代价的。索引需要空间，并且如果数据库中的数据在不断更新，索引也需要相应地调整。当发生在线事务时，调整索引的开销就会变得非常大。

因此应该有一个折中。虽然在访问数据时利用索引可以节省大量的时间，但是索引的创建和维护也带来了附加的处理和数据存储开销。

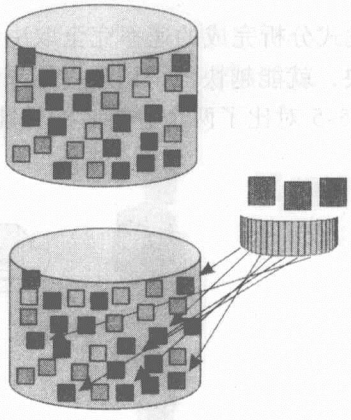


图15-6 在有索引和没有索引的情况下寻找数据

15.9 移除休眠数据

提高数据仓库性能的第二个好方法就是移除休眠数据。休眠数据是指那些不再被访问或访问概率很低的数据，它由于妨碍整个系统的运行而严重地影响了系统性能。它就像人身体中的胆固醇。胆固醇含量越高，心脏为了维持正常的身体机能而搏动得就越强烈。胆固醇含量越少，循环系统的工作也就越有效率。

每个计算机系统都有一定量的休眠数据。只有在存在非常大量的休眠数据时，系统的性能才会受到损坏。

图15-7描述了两个数据库。上面的数据库包含了大量的休眠数据，下面的数据库则只含有极少量的休眠数据。因此，访问下面的数据库要比访问上面的含有大量休眠数据的数据库要高效得多。

这样说来，把休眠数据从数据库环境中移除就是一个很好的手段。虽然移除休眠数据对交互区已经是可行的了，但与整合区更是非常相关。

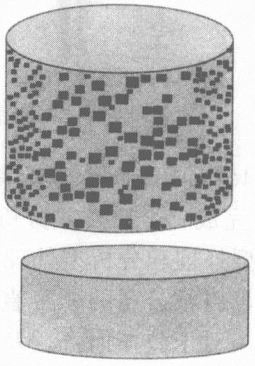


图15-7 访问没有休眠数据的数据库要比访问有休眠数据的数据库有效得多

15.10 终端用户培训

在提高性能上还有另一个简单的想法也能产生很好的结果。这个想法就是在允许终端用户使用数据仓库之前对他们进行培训。在终端用户培训上有两个方面需要强调：

- 数据仓库中有什么数据，这些数据的结构和格式又是怎样的？
- 怎样创建高效的数据查询？

少量的用户培训对保证数据仓库的高效使用和处理性能是有很大帮助的。

终端用户培训在整个DW2.0环境下都是适用的，尤其是在整合区和归档区。

15.11 监控环境

在整个DW2.0环境下实现较好性能的另一项技术是监控环境。令人惊异的是，很多人都没有对重要的数据库和技术基础设施进行监控。当问题发生时，性能监控是一个极好的用于检测和诊断问题的工具。而当问题不能诊断时，必要的补救措施也只是猜测。

与DW2.0环境相关的有两种监控：交互区的事务监控和整合区的数据仓库监控。

事务监控监测事务处理的速度、事务处理中所用的资源和事务在等待队列中的等待时间。数据仓库监控查看休眠数据及用户正在访问的数据。图15-8描述了数据仓库活动和数据的监控。

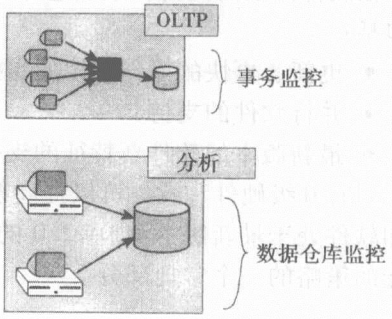


图15-8 环境监控是提高性能的一种重要方法

15.12 容量规划

容量规划问题很自然地伴随着事务性能和数据使用的监控而出现。容量规划的目的是主动预测可能出现系统性能变差的时间，这样就可以在其发生之前便采取补救措施。

如果没有容量规划，直到容量消耗完前系统性能可能会一直很好。然而一旦消耗完容量，由于获得新设备以及技术升级不可能很快完成，因此性能变差以致整个组织都要遭受损失，直到获取并实现了更大的容量。因此，在消耗完数据仓库的容量之前，主动进行容量规划会使组织获得最佳的利益。

虽然容量规划在整个DW2.0环境下都很重要，但与交互区尤为相关。图15-9解释了数据仓库性能监控是如何对主动的容量规划提供特别有用的预测信息的。

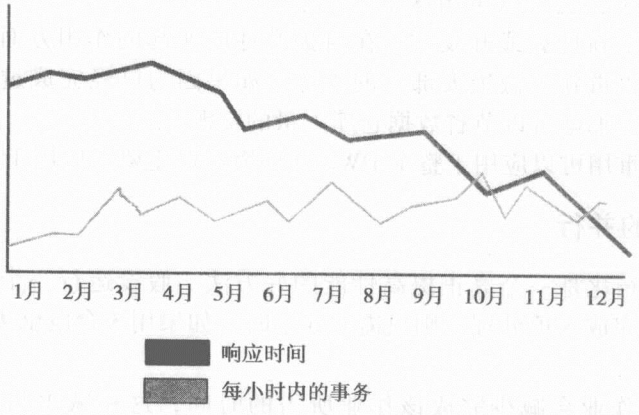


图15-9 容量规划使组织在表现其性能时处于一种前瞻性的状态

容量规划总会引出硬件和软件升级的问题，例如，经常会出现运行在DW2.0环境下的某一部分硬件或软件容量超过限度的问题。在这种情况下，需要补充更多新的资源到这个环境中来。数据仓库硬件容量升级涉及硬件环境下许多方面的升级，比如：

- 更大的主存

- 更大的缓存
- 更快的内部速度
- 存储管理的并行化
- 额外的数据存储

在硬件升级的同时，数据仓库容量的增加还涉及软件的升级。软件升级的一些典型特征包括：

- 更新、更快的操作系统的支持
- 并行软件的支持
- 最新版本的软件新特征的支持

同时升级硬件和软件可以提高系统的性能，有足够多的理由可以说明这一点。保持硬件和软件处于最新版本和 DW2.0 的每个部分都是相关的。系统升级是维持数据仓库优良性能的策略的一个常规部分。

15.13 元数据

一个可靠的元数据基础结构是好的数据仓库性能的一个必不可少的组成部分。元数据基础结构描述了数据在 DW2.0 环境中驻留的位置。

图 15-10 描述了 DW2.0 环境下的元数据。

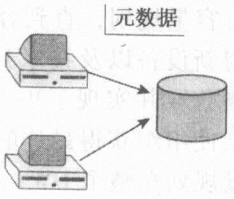


图 15-10 元数据基础结构对于系统性能是很重要的

元数据和性能之间也许并没有明显的关系，但事实上元数据和系统性能有着实际且积极的联系。元数据是可重用性的关键。如果没有可重用性，每当出现一个困难或问题时，一切都必须从头做起。而有了可重用性，当一个组织已经产生一个结果或完成一个分析时，就可以重用这个结果或分析而不用再重新计算或开发它。在对提升性能所起的作用方面，绝对没有任何事情可以比得上不用再重新去做绝大部分的工作。如果因为已经完成过一次主要的分析任务而不用再去做它，那就可以节省数据仓库大量的资源。

对现存分析的重用可以应用于整个 DW2.0 环境，但主要是应用于整合区和归档区。

15.14 批处理的并行

处理过程的并行化是一个真正提高性能的好方法。假设运行一个作业需要 12 小时。如果用两台电脑去完成这项处理，则只需要 6 小时。如果用 3 台电脑去完成这项处理，则只需要 4 小时。

并行处理一个作业会减少完成该作业所需的时间，这种减少与处理器的数量成比例。如果用两个处理器，所需时间会是原来时间的 1/2。如果用 10 个处理器，则时间就会降低到原来的 1/10。如果作业可以被分配到 n 个处理器，则所用的最大时间就会是原来的 $1/n$。

必须注意到，虽然许多作业是可以被并行处理的，但还有些作业是不能够被并行处理的。

作业的并行化作为提高性能的一种方法被应用到整个DW2.0环境。图15-11描述了在DW2.0环境下，并行减少了处理所消耗的时间。

在图15-11中，并行应用于多个没有关联的机器上。这当然是实现高吞吐率的一种方法，但还存在其他也同样很有用的并行化形式。

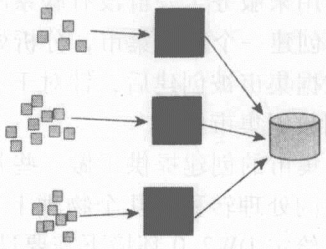


图15-11 批处理作业流的并行可以减少消耗时间

15.15 事务处理的并行

事务处理的并行工作并不是只针对作业的处理。当事务处理中发生并行时，事务被合并并且集中管理，但是数据和用来操作处理的处理能力是分开管理的。管理这种类型处理的操作系统被称为“无共享环境”。在这种无共享环境下，每个处理器拥有并且管理属于自己的数据，每个事务被完全和其他事务的执行区分开来，这样就产生了极高的吞吐率。这种形式的并行按惯例一般出现在DW2.0中的交互区。图15-12就是一个无共享的事务处理环境示意图。

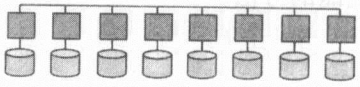


图15-12 存储管理的并行是另一种方式

15.16 工作负荷量的管理

系统要达到一个较好和敏捷的性能，通过系统的工作负荷量是一个很重要的因素。为了理解性能和系统承受的工作负荷量之间的关系，我们可以考虑一条道路和其上运行的车辆之间的关系。现在考虑一个问题：一辆保时捷在道路上能够跑多快呢？除了极少数例外，答案并不是每小时185英里。保时捷只能以它前面的运输工具的速度行进。如果是在墨西哥城并且是高峰时间，保时捷就只能以每小时2英里的速度前进。如果是在德国高速汽车专用公路，那么保时捷的速度就可能达到每小时185英里。但是，如果交通堵塞并且路上有大量半拖车，那么各种类型的车辆所能达到的速度都不会很高。

事务处理流也是这样工作的。事务的类型、事务的大小、共享相同处理程序的事务数量，都对整个数据仓库所能达到的速度有直接的影响。

交互区只允许执行少量的事务，这也就意味着更快的执行速度。而整合区允许不同大小的事务混合在一起执行，也就是说对于整合区，一般期望得到一个综合响应时间。在归档区中，事务一般都很大，因此归档环境下的响应时间性能一般都较差。

图15-13说明了工作负荷量的构成和响应时间的关系。

从战略的角度来看，数据朝向数据集市移动能够极大地提高数据仓库的性能。

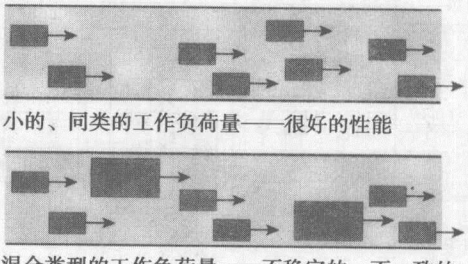


图15-13 当要达到一致的好的性能时，工作负荷量的构成是一个大的因素

15.17 数据集市

数据集市是用于满足一组用户的分析需求的数据集合。一般情况下，数据

集市是依据不同的组织而建立起来的，例如，金融、销售和市场数据集市。数据集市经常被设计用来服务于一群没有联系的人，但他们基本上都使用相同的方法查看数据。

通过创建一个数据集市，分析处理就可以从一个环境转移到下一个环境。换句话说，当一个数据集市被创建后，针对于 DW2.0 环境的处理的数量就会减少，其中一些处理就会转移到数据集市中。

数据集市的创建提供了做一些其他的对于数据仓库性能十分有益的事情的机会。把数据集市中的处理转移到几个物理上相互分离的处理器可以显著减少公司数据仓库中处理的成本。给定 DW2.0 环境下需要进行的处理数量，用于支持这些处理的处理器可能是十分昂贵的。把分析循环移出 DW2.0 企业数据仓库环境可以减少处理成本，因为相比起来数据集市通常需要少得多的硬件支持。因此，除了提高 DW2.0 的性能外，这也是一个很强的建立数据集市的动机。

数据集市几乎仅应用于 DW2.0 的整合区。图 15-14 将数据集市的创建描述为一种提高性能的方法。

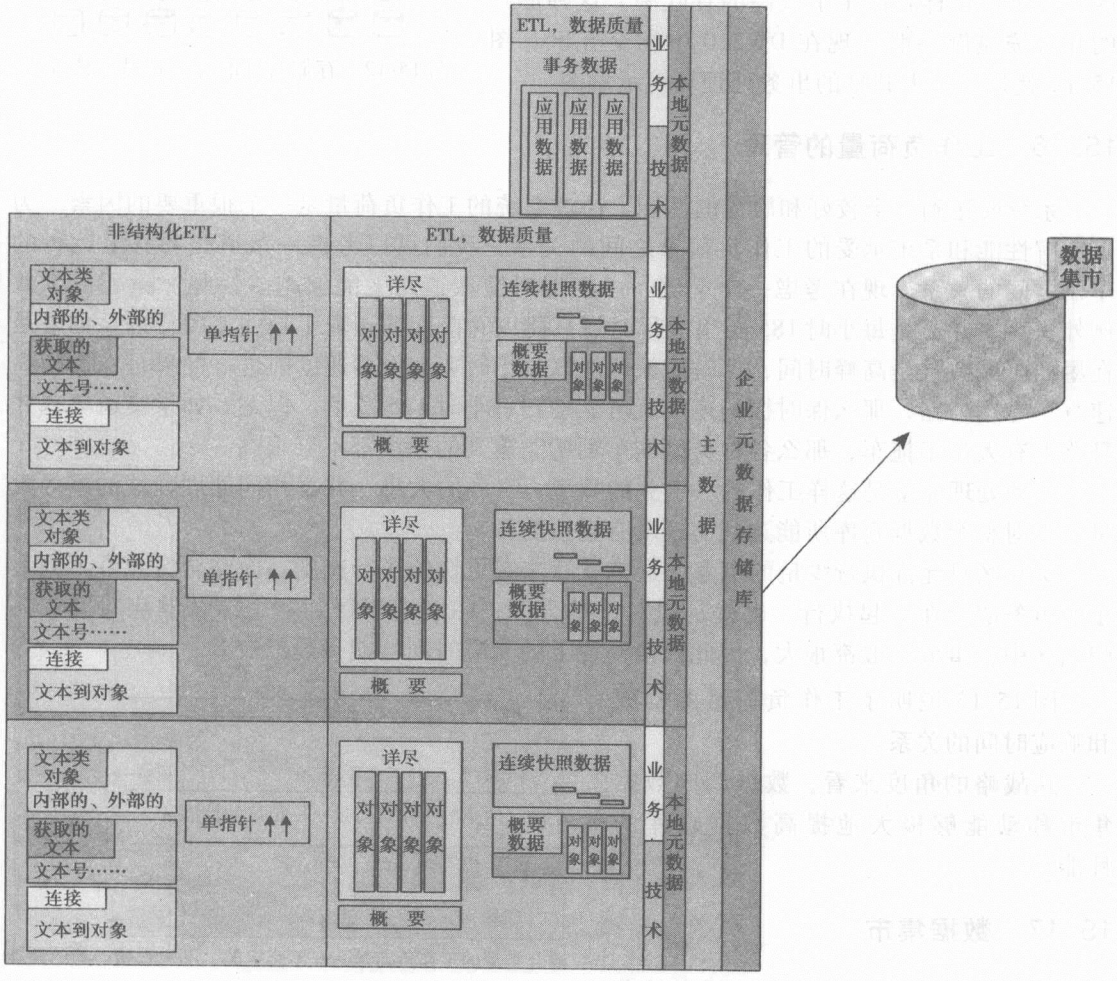


图 15-14 将数据移动到数据集市是增强性能的另一个好方法

15.18 探索工具

探索工具以和数据集市非常相似的方式来提高 DW2.0 的性能。图 15-15 描述了探索工具的创建可以提高 DW2.0 企业数据仓库环境的性能。

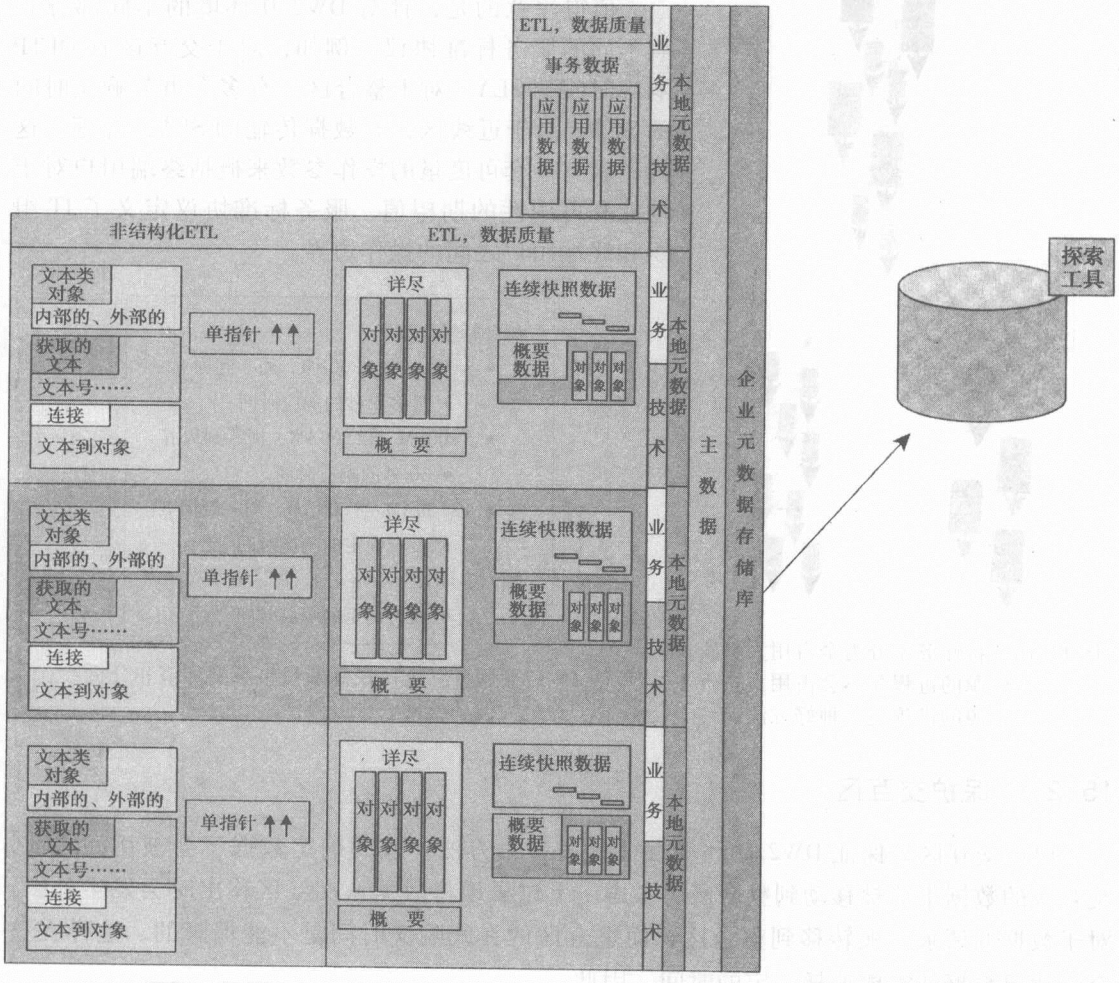


图 15-15 将数据移动到探索工具是增强性能的另一个好方法

15.19 将事务分为不同的类

将事务分为不同的类是提高数据仓库性能的另一种好方法。图 15-16 显示了将事务分成两类。

关于事务分类技术的一个有趣的问题是：一个组织如何决定一个事务将会是快还是慢？通常，事务要访问的数据越多，速度就会越慢。如果事务将访问整个数据库，那么速度将会很慢。而如果事务只访问一部分数据，那么将会在较短时间内完成。

15.20 服务标准协议

一旦事务的执行速度被确定后，管理该事务执行的一种方法就是创建所谓的“服务

标准协议”（service level agreement，SLA）。

SLA是对于服务的所期待标准的一种声明。一般情况下，SLA是针对事务响应时间和系统可用性的。

图15-17显示了一个服务标准协议。

值得注意的是，针对DW2.0环境的不同部分有着不同的服务标准协议。例如，对于交互区有OLTP响应时间的SLA。对于整合区，有多个事务响应时间的SLA。对于近线区，有数据传输的SLA，等等。这样，就有一套可度量的操作参数来概括终端用户对于性能和可用性的期望值。服务标准协议定义了IT组织和终端用户之间的操作边界。

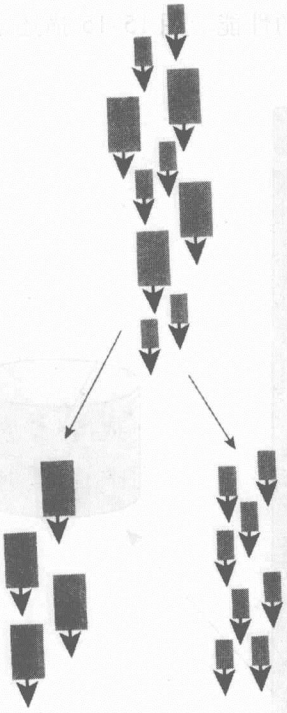


图15-16 将任务流分为全占用大量资源的进程和不会占用大量资源的进程是一种好方法

服务标准协议
- 8:00AM~5:00PM，周一到周五
  - 最多5秒的响应时间
- 5:00PM~12:00AM，周一到周五
  - 30秒的响应时间
- 12:00AM~8:00AM，周一到周五
  - 没有确定的响应时间

周六、周日
- 没有确定的响应时间

图15-17 服务标准协议是衡量事务型环境和分析型环境的一种好方法

15.21 保护交互区

保护交互区是保证DW2.0环境性能的另一个方法。直到确定数据不会被访问之前，交互区的数据不能被移动到整合区。考虑一下过早地将数据从交互区移出时会发生什么。对于数据的请求会被转移到整合区，而整合区的在线响应时间是不能保障的，这样在整合区找寻数据就需要消耗一定的时间。因此，如果还有访问交互区数据的可能，就应该把这些数据保留在交互区。

图15-18说明了当还很有可能访问交互区的数据时，数据就应该保留在交互区。

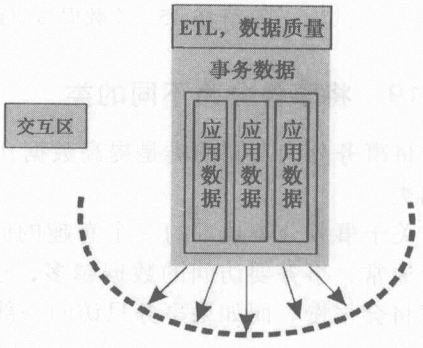


图15-18 当还很有可能访问交互区的数据时，保证数据一直不会离开交互区

15.22 数据分割

数据分割要求在存储数据时，将数据在物理上分割为多个离散的部分。数据分割保证了不同的数据集可以分开处理，这样极大地增强了数据的灵活性。

为了说明物理上将数据分割所带来的优点，考虑一个10TB大小的数据库。某一天需要增加一些数据或者重新计算数据库中的一小部分数据。当这项活动发生时，整个数据库可能会变得不可用。现在考虑如果将数据库分割为1TB的块时会发生什么。只有一小部分数据在修改过程中是不可用的。这样做的话，就可以自由地使用大多数数据来完成一些之前无法完成的处理要求。

分割方法适用于DW2.0环境下任何存储着大量数据的部分。图15-19描述了这种数据分割。

15.23 选择合适的硬件

在DW2.0环境下，选择合适的硬件和软件对性能有很大的影响（见图15-20）。

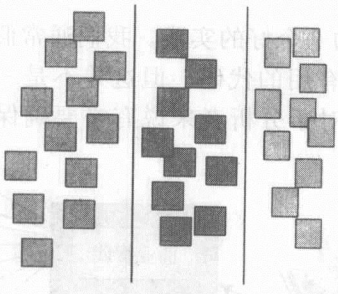


图15-19 存储时的数据分割

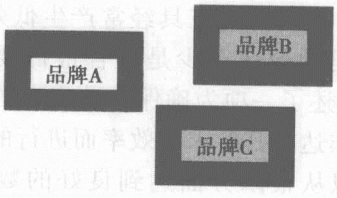


图15-20 选择满足性能要求的合适的技术

15.24 区分“农民”和“探索者”

基于数据不同的使用方式来区分处理工作也是优化性能的一个方法。为此，用户可以大体被分为两类——“农民”和“探索者”。

农民是指从事的分析活动有着规律性和可预见性的终端用户。他们在开始查找数据之前就知道他们想得到什么。而探索者的活动则很难预测，探索者可能6个月内不会做任何分析活动，但突然在一周之内需要做大量的分析型处理。他们在开始查找之前并不确切地知道他们想要什么。

农民和探索者都是DW2.0环境的合法用户。他们都应得到资源，并且都为公司的决策过程做出了颇有价值的贡献。但是，有一个很好的原因使得需要将农民和探索者分开，即这两种类型的数据仓库使用者进行的分析活动是完全不同的，就像油和水。当将两种不同类型的用户群体的分析活动分开时，整个数据仓库环境的性能就提高了。图15-21更加说明了在相同的环境下农民和探索者不能混合在一起，而需要被区分开。

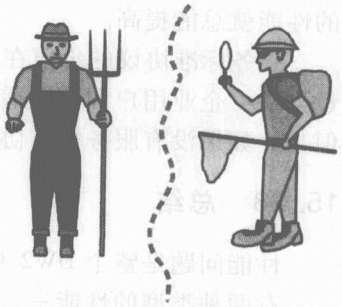


图15-21 将农民与探索者分开

15.25 数据的物理分组

为提高数据仓库的性能，另一个应该完成的基本活动就是：当大多数用户对数据进行分组使用时，将这些数据在物理上也进行分组。

图15-22通过颜色深浅区分了五种不同类型的数据。如

果分析者认为95%的情况下这些数据都是一起被访问的，那对于数据库设计者来说，将这些数据以其最常见的访问方式物理地组合起来是很有意义的。以这种方式将数据组合起来，系统就可以有效地获取数据。

像这样根据数据被访问和使用的形式来组合数据被认为是数据的非正规化。通常，数据的非正规化对于公司数据仓库来说并不是好的数据架构实践，并不推荐。公司的主要数据仓库中的数据不应该以这种方式非正规化，除非确定了一个极其有必要的、一致的和持久的数据子集访问方式。

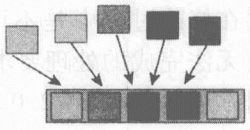


图15-22 当数据被合并起来使用时，它们应该在物理上组合起来，虽然这种组合是非正规化的

15.26 检查自动产生的代码

检查由分析工具产生的代码是提高数据仓库性能的一个好的实践。我们通常假设分析工具（如商业智能工具）可以并总是自动产生高效且有用的代码。但这并不是一个安全的假设，因为分析工具经常产生低效的代码。因此，对于分析者来说有必要确保正在产生的代码的效率至少是符合最低要求的。图15-23描述了一项为确保自动产生的商业智能查询操作达到其要求的效率而进行的检查。

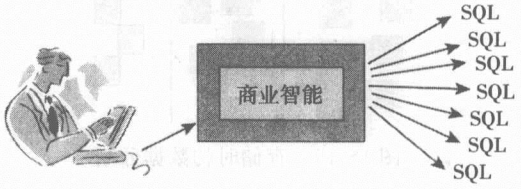


图15-23 不要假定所使用的分析工具产生的码是有效甚至是正确的

可以从很多方面达到良好的数据仓库性能。以上所讨论的措施几乎都可以在任何时刻同时实现。

15.27 企业用户的观点

企业用户对整个DW2.0中活动的性能都有敏锐的意识。他是通过对各项性能的交付与否来了解各性能的。

企业用户的职责是，在系统性能出现衰退时通知系统管理员。通常会有服务标准协议，它能够加强或减弱企业用户与系统管理员之间的对话。

企业用户很少参与对性能的补救工作。关注DW2.0环境内部出现的问题是系统管理员的职责。

有时，企业用户会要求一个新级别的性能。例如，可能会在归档环境中要求在线响应时间。只要企业用户愿意为性能埋单，就确实可以提升性能的级别。不过，大多数企业用户并不愿意为系统资源和服务付费。如果企业用户愿意对技术升级进行投资，那系统的性能就总能提高。

服务标准协议的价值在这一点上就表现得十分明显。有了服务标准协议及其附属的度量方法，企业用户和系统管理员就可以进行富有成效的商谈，从而避免许多实际问题。但是，如果没有服务标准协议，企业用户和系统管理员就很难进行明智的对话。

15.28 总结

性能问题是整个DW2.0环境一个必不可少的特征。

有两种类型的性能——事务型和分析型。

当事务型处理的性能出现问题时，公司的操作型活动会受到影响。而当分析型处理的性能出现问题时，公司的分析能力就会受到影响。

良好的性能是多方面因素共同作用的结果，包括但不仅限于以下几个方面：

- 选择合适的索引。
- 尽快移除休眠数据。
- 培训终端用户怎样识别好的和差的代码。
- 监控事务和数据仓库环境，以便当性能变差时，可以有一个用于判断到底出现什么错误的起点。
- 规划容量以便组织可以预见资源将要被用完。
- 升级，保证正在使用的是最新版本的硬件和软件。
- 元数据，以便利用重用性，最小化所需的工作量。
- 批处理，减少消耗的时间。
- 事务并行，有效地处理大的工作负荷量。
- 工作负荷量管理，保证一项工作不会因为大小而和其他工作冲突。
- 数据集市，完成从中央数据仓库中转出的主要的分析型处理。
- 探索工具，将统计型处理移动到其他位置进行。
- 基于事务所要使用的资源将事务分为不同的类。
- 服务标准协议，建立量化的指标来衡量性能。
- 保护交互区来最小化资源的争夺。
- 将数据分成不同类别来分别管理。
- 选择合适的硬件和软件来实现性能。
- 区别农民和探索者的工作。
- 非正规化数据，将经常会被同时访问的数据物理地放到一起。
- 检查由工具（如商业智能工具）自动产生的代码。

这些只是在DW2.0环境下提高性能的一些技术和方法。

第16章 迁 移

DW2.0是一个巨大且复杂的环境，需要利用大量的资源并经过很长的时间来建立它。图16-1指出DW2.0环境更像一座城市而不是一座房子。

16.1 房屋和城市

建一座房子需要相对较短的时间，在建立过程中有一个明确的起点和终点。房子一般是在某个单一的时间点达到了可用性，即在一个点它还是不可用的，而在另一个点它就可用了。

图16-1 构建DW2.0环境时，你是在建一座城市而不是一座房子

而一座城市的建造过程是非常难的，并且需要很长一段时期。从城市里的第一座建筑物建起来开始，城市就可用了。城市可能有规划也可能没有规划，即使各个城市都有一些相同的特征，但各个城市还是拥有各自的特征。例如，雅典、罗马、纽约和东京都有飞机场、市政大楼、住宅区和高档区，但是它们之间却不会被混淆。雅典有巴特农神殿，巴黎有埃菲尔铁塔，纽约有金融区，东京有横跨在横滨海湾上的螺旋交通大桥。

上述规则同样适用于DW2.0数据仓库，虽然使用了相同的结构，但可口可乐、花旗银行、帝国商业银行以及克莱斯勒等企业的DW2.0的实施是非常不同的。

如果企业几乎从来没有打算建立一个DW2.0环境，那又怎么最终实现DW2.0环境呢？答案就是会逐渐发展。随着时间的推移，企业逐渐向DW2.0构架迁移。

16.2 在一个完美情况中迁移

在一个完美情况中，DW2.0的构造过程模拟了DW2.0环境中数据流的流动过程。图16-2反映出一个“完美世界”中DW2.0构架的实施过程。该图显示了在没有其他数据仓库存在的前提下，构造DW2.0数据仓库的步骤。每一层处理的建立都作为下一层处理建立的基础。

16.3 完美情况几乎永远不会发生

但是图16-2中的顺序只是一种理论上的构造顺序，一个DW2.0数据仓库几乎从来不会以所描述的那种自顶向下的顺序来构造的。DW2.0数据仓库不能以这种“自然”的顺序构造的主要原因是，几乎所有创建DW2.0数据仓库的人都在适当的位置拥有了一个已存在的数据仓库。

图16-3给出了几乎所有人在构造之初的基础结构，包括遗留的应用环境、ETL处理、数据库或者数据仓库。这是大多数企业的基础结构中最基本的组件。

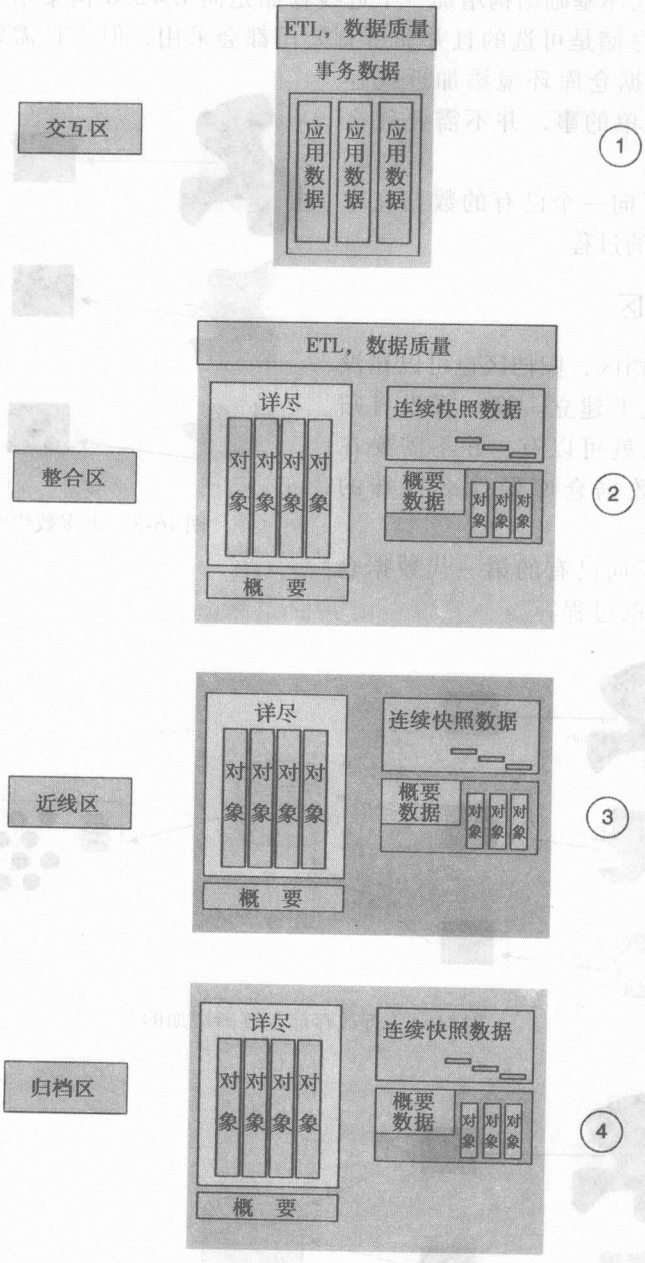


图 16-2　DW2.0 建立的“自然”顺序

16.4　增量式添加组件

关于 DW2.0 构架的一个好的消息是，它的大部分组件都可以根据需要，独立的、增量地添加进来。这种独立的、不断增加的能力意味着公司可以以一种有序的方式向 DW2.0 环境迁移和进化。迁移并不是根除并丢弃已有系统，相反，在构建 DW2.0 时，DW2.0 的基础结构组件可以建立在已有的数据仓库上。

给现有的数据仓库基础结构增加一个近线存储是向 DW2.0 构架增量迁移的一个很好的例子。虽然近线存储是可选的且并非所有公司都会采用，但一旦需要，它就是不可替代的。向第一代数据仓库环境添加近线存储在构架上是件简单的事，并不需要什么特别的工作或准备。

图 16-4 说明了向一个已有的数据仓库环境添加近线存储的过程。

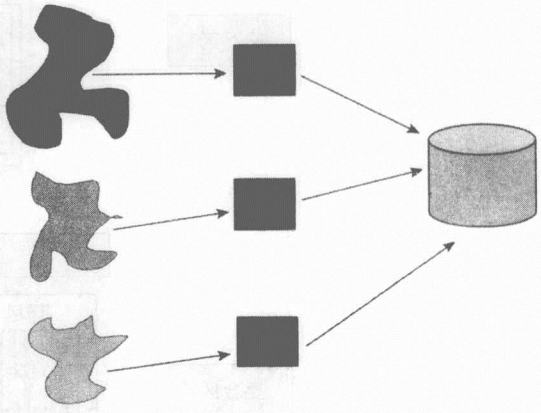


图 16-3 大多数组织开始的地方

16.5 添加归档区

接下来考虑归档区，归档区也可以在没有预先准备的情况下建立。第一天没有归档设备，但第二天就可以有，并不需要在过程中对第一代数据仓库做什么特殊的事情。

图 16-5 说明了向已有的第一代数据仓库环境添加归档区的过程。

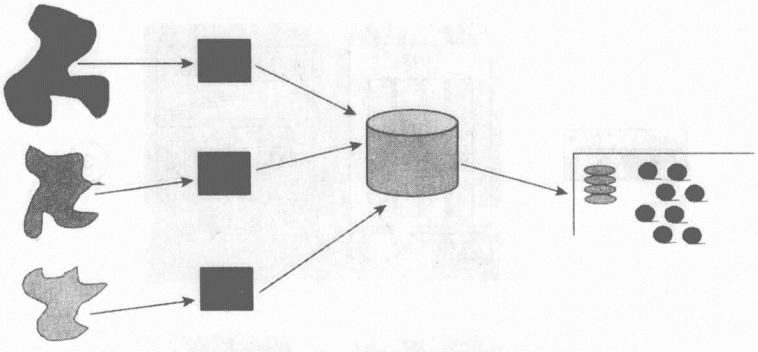


图 16-4 近线存储是逐渐增加的

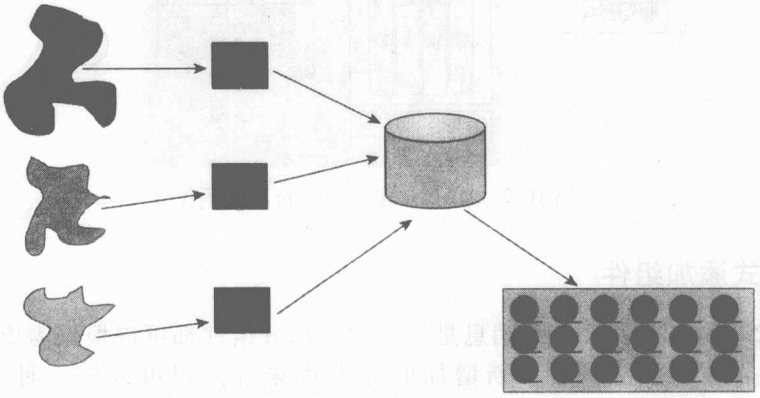


图 16-5 添加归档区是相对容易的

16.6　建立企业元数据

对于 DW2.0 元数据设备也是与如上同样的考虑。通常，本地元数据已经被存放在某个适当的位置了。无论是否使用本地元数据，提供技术的经销商通常都会提供元数据（例如 ETL 元数据、商业智能元数据和 DBMS 元数据）的本地存储和管理的设备。所以本地元数据通常已经存在，需要添加的是企业元数据。建立企业元数据通常由以下三个步骤组成：

- 建立企业元数据存储库。
- 将本地元数据移入企业元数据存储库中。
- 调整本地元数据以适应企业元数据的格式。

最后一步通常总是最难的，调整本地元数据使其遵循全局的、企业的格式和结构是一项非常困难的任务。

16.7　建立元数据基础结构

建立一个企业级的元数据存储库决不需要破坏或者丢弃原有的环境，相反，DW2.0 的元数据基础结构是建立在已存在的数据仓库的基础结构之上的。

图 16-6 描述了在已有的第一代数据仓库上建立企业元数据基础结构的过程。

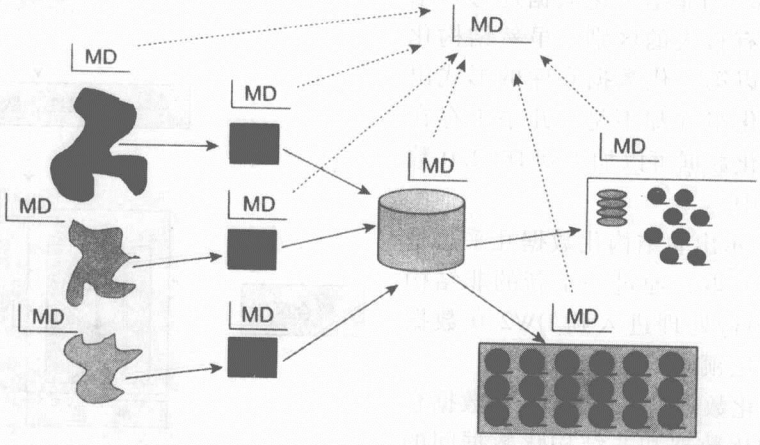


图 16-6　从各种来源收集起来的元数据构成企业的元数据存储库

16.8　“吞没”源系统

在已有的操作应用环境中，如果任意一个位置可能已经停止运行，那么它就是一个遗留应用程序，需要被 DW2.0 环境的交互区来消化吸收。在大多数情况下，交互区会“吞没”旧的源系统，而在其他情况下，源应用程序会保持原样，并简单地继续向交互区提供数据。

在源应用被交互区吞没的情况下，这个应用肯定就是一个陈旧过时的遗留系统了。这些被吞没的遗留应用是很久以前设计的，用来满足当时的业务需求，而这种需求早已改变了。如果交互区没有随着改变，那么这些遗留应用无论如何都需要再重新改造。

图 16-7 显示了如何将一些遗留应用吸收到交互区中。

16.9 作为缓冲器的 ETL

ETL 处理扮演着一个整个数据仓库演进和迁移过程的缓冲器的角色。通过 ETL 转换后，在操作源应用程序世界产生的大的变化对交互区的影响将会降到最小。同样，交互区也可能出现一个大的变化，通过 ETL 转换后，其不会对整合区产生影响或者降到最小。

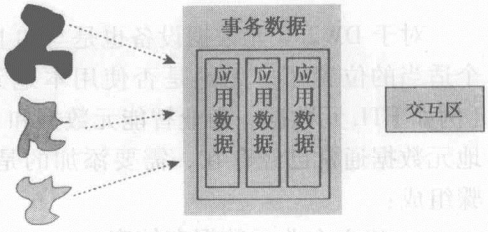


图 16-7 将遗留应用吸收到交互区

图 16-8 给出了 ETL 在不同区之间充当缓冲器的示意图。

16.10 迁移到非结构化的环境

非结构化的数据领域是 DW2.0 数据仓库环境的一个最新、最重要的特征。在许多 DW2.0 环境中，非结构化的数据作为一个添加组件打开了通向更多种类的分析和决策支持处理的大门。

DW2.0 环境的非结构化数据迁移与结构化数据迁移有很大的区别，虽然结构化环境几乎总是以第一代数据仓库的形式出现，但非结构化组件却不是。几乎不存在已有的非结构化数据可以加入到 DW2.0 数据仓库环境当中。

图 16-9 显示出非结构化数据几乎总是从它的文本源获取，通过一个新的非结构化数据 ETL 例行处理进入到 DW2.0 数据仓库的非结构化领域。

在非结构化数据进入到 DW2.0 数据仓库中后，结构化数据和非结构化数据间的链接也就建立起来了。图 16-10 描述了在一个 DW2.0 区域结构化数据域和非结构化数据域的链接的形成。

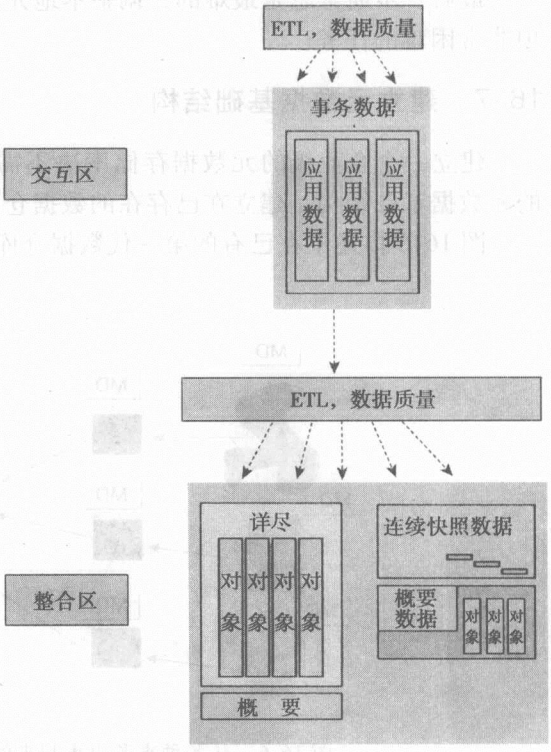


图 16-8 ETL 处理类似一个缓冲器

随着时间的推移，非结构数据也将会停止使用，其会移到 DW2.0 归档区的非结构化数据域当中。在第 19 章我们会详细介绍关于非结构化数据的问题。

16.11 企业用户的观点

企业用户会间接地参与迁移。企业用户决定了 DW2.0 之中应包含哪些新的对象域，数据在什么时间应该进入归档区和近线存储区，数据如何从一个区向另一个区转换。

但是最终，企业用户不会涉及 DW2.0 环境建立过程中的数据迁移。

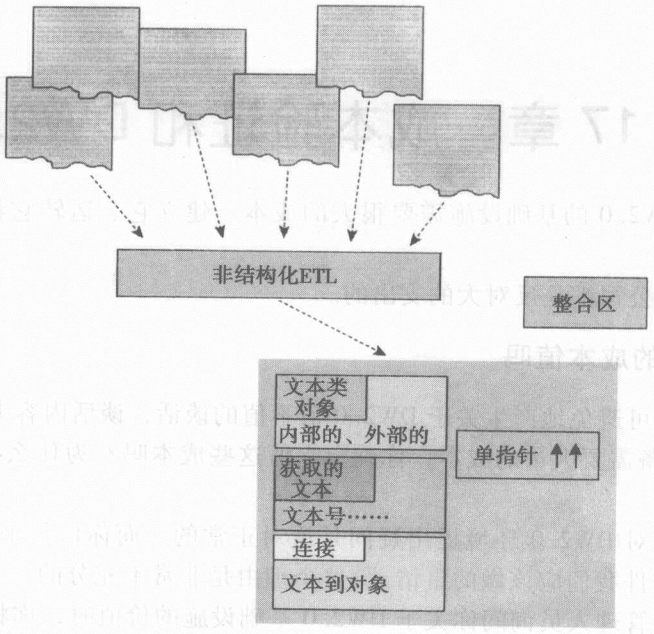


图 16-9　非结构化数据由文本或者其他形式的非结构化数据转换而来

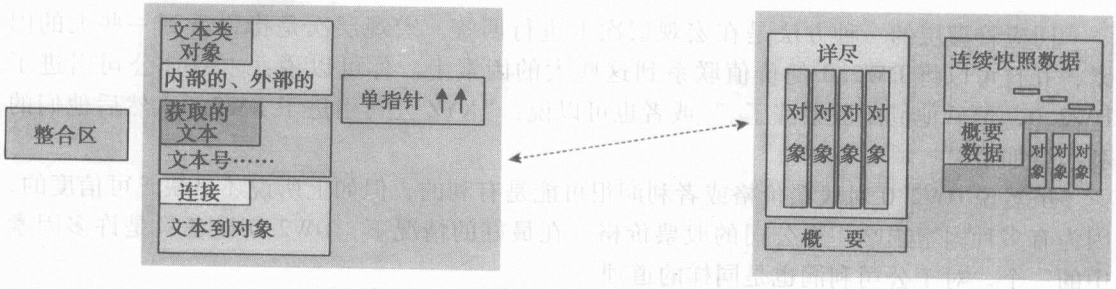


图 16-10　非结构化环境和结构化环境相联

16.12　总结

DW2.0 数据仓库的构架有一个自然的迁移顺序。自然迁移的顺序是根据数据流动方向而定的——首先进入交互区，然后是整合区，然后是近线区，最后是归档区。虽然自然顺序已经定义好了，但它只是理论上的。

在实际中，DW2.0 环境是从第一代数据仓库发展形成的。归档环境可以独立加入，近线环境可以独立加入，同样，企业的元数据结构和非结构化数据域也可以独立加入。

针对不同业务的需要，可以将不同的组件加入到 DW2.0 环境中。

遗留应用系统是仅有的、预先存在的可能在移入 DW2.0 的过程中遭到破坏或者被代替的系统。有时候，旧的系统环境会因为它太过时、太过于脆弱，以至于相比整合数据进入旧系统，还不如重写系统。

第 17 章　成本验证和 DW2.0

毫无疑问，DW2.0 的基础设施需要很大的成本。建立它、运转它都需要成本，所使用的设备也有成本。

然而，大部分公司都是反对大的支出的。

17.1　DW2.0 的成本值吗

很自然，会不可避免地产生关于 DW2.0 值不值的谈话，谈话内容大体如下：

“这个新的设备需要大量的成本，你确定它值这些成本吗？为什么我要做一个这么大的投资？”

高层管理人员对 DW2.0 环境提出疑问是绝对正常的。而你仅是坐在那里说：“我的直觉告诉我这是一件我们应该做的事情。”这个理由是非常不充分的。

所以，当高层管理人员询问你关于 DW2.0 基础设施的价值时，你将如何回答？

17.2　宏观层次的价值验证

回应管理层的一种方法是在宏观层次上进行回答。宏观层次是指你看到一些大的因素，并且可以将 DW2.0 的价值联系到这些大的因素上。你可以说：“ABC 公司引进了 DW2.0，然后他们的股票涨了。”或者也可以说：“XYZ 公司引进了 DW2.0，然后他们的利润增加了。”

虽然说 DW2.0 对股票价格或者利润很可能是有利的，但如上所说还是缺乏可信度的，因为有多种因素影响一个公司的股票价格。在最好的情况下，DW2.0 设备只是许多因素中的一个。对于公司利润也是同样的道理。

DW2.0 环境对这样大的宏观因素的作用似是而非，所以说这种话的人的可信度也受到质疑。

图 17-1 给出了在宏观层次证明 DW2.0 的价值。

17.3　微观层次的价值验证

在微观层次上解决 DW2.0 环境的价值验证问题是一种更为可信的方法。在微观层次上可以对 DW2.0 环境的建立和操作形成更为令人信服的理由。

比如，有两个公司 A 和 B，见图 17-2。

两个公司的信息基础结构非常相似，同时两个公司都有许多旧的遗留系统。公司以旧的面向事务的技术来完成公司的业务，并且已经保持了很长的一段时间。

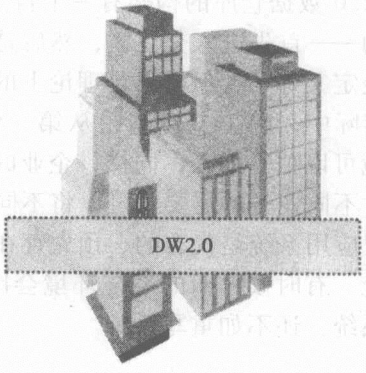


图 17-1　在宏观层次对 DW2.0 进行价值证明是很难的

图 17-3 显示了公司 A 和 B 都有遗留数据并且需要新的分析方法。

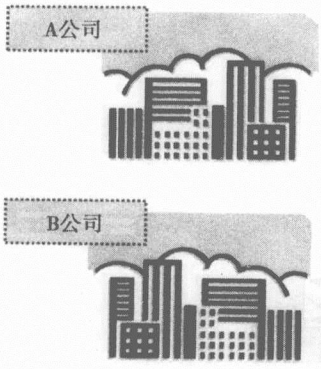


图 17-2 考虑两个公司

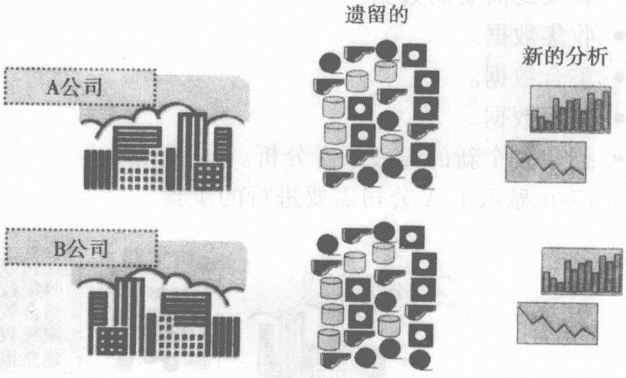


图 17-3 无论是 A 公司还是 B 公司都存在过时的系统并需要新的分析

17.4 公司 B 拥有 DW2.0

A 公司和 B 公司所拥有的基本构架组件几乎都是相同的。但是有一个非常大的不同，就是公司 A 没有 DW2.0 设施，而公司 B 有。图 17-4 描述了两个公司间的这个基本区别。

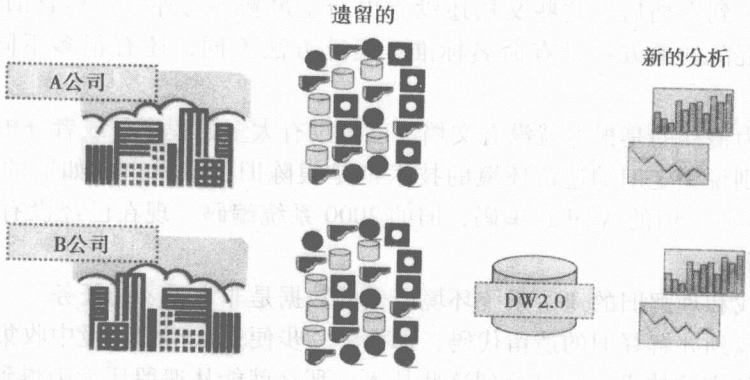


图 17-4 对于两个公司来说，唯一不同的地方就是 B 公司有 DW2.0 环境

17.5 生成新的分析

无论如何公司都需要新的分析。A 公司如何在它遗留的数据基础上建立新的信息？图 17-5 描述了 A 公司需要回到它的遗留应用程序库去生成一个新的分析。

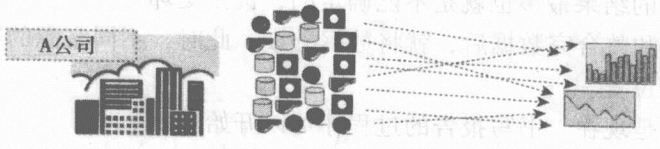


图 17-5 A 公司是如何建立分析的

那些遗留的数据是 A 公司唯一要操作的数据，没有其他的数据。

所以，A 公司如何去生成新的分析？A 公司需要进入遗留的环境并且

- 查找到需要的数据。
- 收集数据。
- 整合数据。
- 演绎数据。
- 创建一个新的报表或者分析。

图 17-6 显示了 A 公司需要进行的步骤。

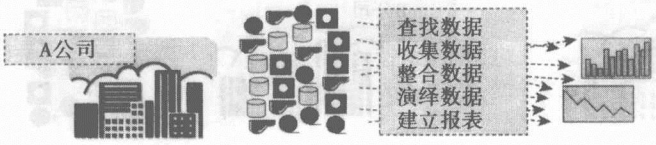


图 17-6 如果要建立新的分析需要做什么工作

17.6 按步骤执行

这些是正常的自然的步骤，现在让我们来考虑执行这些步骤具体都需要什么。

为得到数据，旧的遗留数据就必须有文档记录，然而许多较旧的遗留系统并没有文档。其次，当找到文档后，这些文档还没有更新至最新。另外，在检查旧的遗留系统时还会发现很多混乱的地方：没有命名标准，运算方法不同，还有很多不同的编码标准，等等。

其中，更为糟糕的是根本就没有文档。已经没有太多的程序员或者分析员能够理解当前的技术，更别说锁定旧的遗留环境的技术都是很陈旧的技术。比如旧的 IMS 编码，旧的 Model 204 编码，旧的 Adabas 编码，旧的 2000 系统编码。现在已经没有任何工作者能够明白这些编码。

所以，寻找和理解旧的遗留系统环境产生的数据是非常重要的任务。

假设能够找到并解释旧的遗留代码，那么下一步便是从遗留环境中收集数据。这需要已经消失很久的定位技术。一旦找到这些技术，那么就能从遗留环境中得到大量数据。

接下来是整合数据。如果数据仅来源于一个系统，那么就没有什么问题，但是如果数据来源于多个不同系统，那么就会出现整合的问题。关键字结构不同，数据格式不同，一些数据丢失并且必须采用默认值，相同的属性有不同的名字，数据的定义不同，等等。一个处理旧的遗留数据的人需要去合并多个系统的数据，而这些系统并不是为了合并而设计的。如果没有合适的文档，并且使用了某种神秘的技术来编写系统，那么对系统做一个适当的整合的结果最多也就是不能确定的，甚至更坏。

在找到、收集和整合完数据后，就将数据分级。此时，不同来源的数据将被收集在一个单一的物理存储位置。

现在，也只能是现在，书写报告的过程才可以开始。

在给出的上述这些挑战下，生成一个新的分析需要多少成本？这全部取决于遗留环境以及分析的要求。有些遗留环境实在是过于混乱，而有些就比较好处理；有些报表和分析非常简单，而有些就非常难。

17.7 总成本是多少

在图 17-7 中显示了建立一个分析需要的一系列成本。

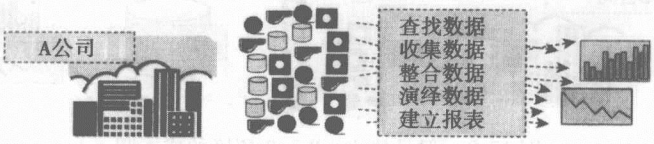


图 17-7 做这个项目需要花费多少钱和多少时间？从 100 000 美元到 10 000 000 美元，从 3 个月到 3 年

根据生成过程中的细节，建立一个新的分析大概需要花费 100 000 美元到 10 000 000 美元，时间大概需要 3 个月到 3 年。由于多种因素，成本往往会发生较大变化，比如：

- 遗留程序的个数。
- 遗留系统的复杂性。
- 遗留应用程序的文档。
- 遗留环境的技术构架。
- 新数据需求的复杂度。
- 需要分析的数据量。
- 新信息所需的数据元素的个数。
- 遗留数据库与当前最新版本的匹配程度。
- 遗留环境的操作系统。

17.8 考虑公司 B

现在再考虑公司 B，它也需要一个新的分析。那么利用 DW2.0 设施去建立一个新的分析需要多少成本？图 17-8 给出了答案。

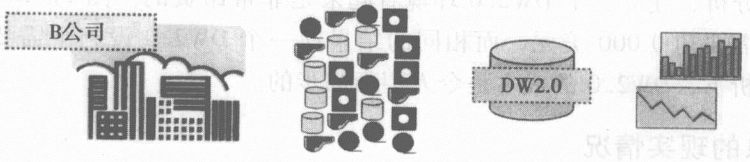


图 17-8 一旦 DW2.0 环境搭建起来了，做这个分析需要花费多少钱和多少时间？从 1000 美元到 10 000 美元，从 1 小时到 5 天

在图 17-8 中可以看到大概需要 1000 到 10 000 美元和 1 小时到 5 天来完成这个分析。从这些数据可以看出公司 B 与公司 A 相比有着非常轻松的信息设施构建过程。而两个公司唯一的不同就是 DW2.0 设施的存在。

根据这些观察值可以看出，DW2.0 极大地降低了一个公司的信息成本，换种说法就是，DW2.0 打开了通往之前本来已经存在却不能访问的信息的大门。

17.9 考虑 DW2.0 的成本

但是仅仅比较关于信息的数字难免会产生偏差，偏差来源于没有把 DW2.0 设施的成

本加入到计算公式。当计算时加入 DW2.0 设施的成本时会发生什么呢？图 17-9 显示了这个公式还需要更多的工作。

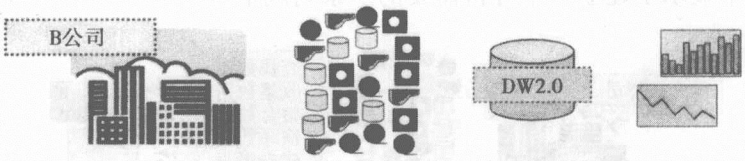


图 17-9 但是建立 DW2.0 环境的成本呢

构建和操作 DW2.0 设施并不廉价。当然大部分成本取决于数据量、用户量、旧遗留系统的数目、数据保存的时间长度以及分析的类型，等等。但是针对此次分析的目的，我们假设一个 DW2.0 设施需要花费 5 000 000 美元。

现在，A 公司生成一个报表的成本是多少？我们假定它需要 700 000 美元去生成一个报表，那么对于 B 公司来说生成相同的报表的成本又是多少呢？是 10 000 美元。

那么现在在这些报表成本中加入 DW2.0 环境的成本，那么成本公式会变成什么样子呢？图 17-10 显示了这些统计数字。

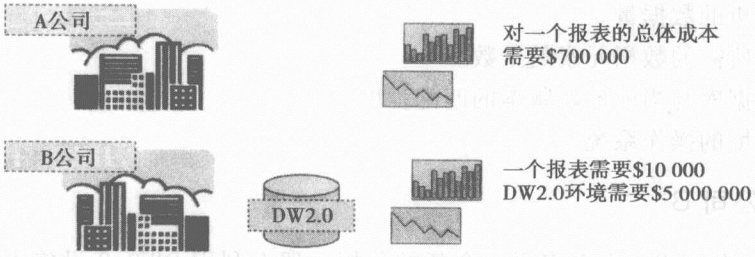


图 17-10 如果我们只需要一个报表会怎样

利用这种分析，建立一个 DW2.0 环境看起来是非常昂贵的。图 17-10 中的数字表明生成一个报表需要 700 000 美元，而相同的结果在一个 DW2.0 设施下需要 5 010 000 美元，在这种分析下，DW2.0 的成本是令人望而却步的。

17.10 信息的现实情况

但是图 17-10 的情况非常不现实。这个图是假定公司只需要一个报表。但是任何一家公司，不管它的规模大小和复杂程度如何，都不会仅在一个报表上运作其所有业务的。即使一个中等规模的公司，至少也要有 100 个报表。财政部门需要一种形式的报表，市场部门要求另一种形式的报表，销售部门要求的形式又不同。而对于大公司来说，就不止是几百份报表了，而是上千份。但是为了我们分析的目的，假设需要 100 份，那实际需要的报表数会怎样改变经济效果呢？

图 17-11 显示了考虑公司实际需要的报表数量的结果。就公司 A 来说，报表的成本为 100 乘以 70 000 000 美元；就公司 B 来说，报表的成本为 100 乘以 1 000 000 美元，但这里只需要一个 DW2.0 设备，所以 DW2.0 环境的成本是不变的。

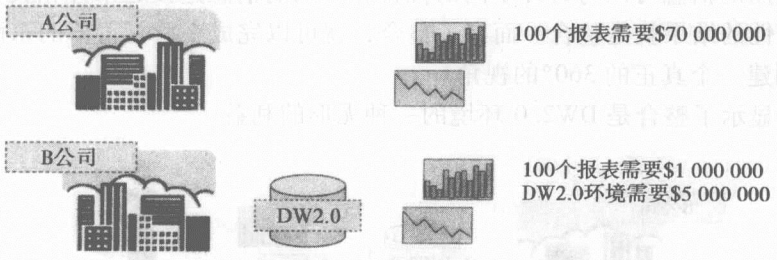


图 17-11 如果我们需要 100 个报表会怎样

17.11 DW2.0 真正的经济效益

现在对 DW2.0 环境成本分析的经济上的分析变得明显了。DW2.0 能够将新分析的成本减少一个量级甚至更多。DW2.0 容许一个公司访问和分析以前从未能够分析的数据。但是原始的 DW2.0 的经济分析并没能够说清这一点。

17.12 信息的时间价值

关于信息的时间价值的例子是确实存在的，而 DW2.0 能极大地加快信息获取的速度。

想象一下，你的老板走进你的办公室要一份报表。你在一系列旧的遗留系统中寻找，6 个月后终于找到了那份分析报表。当你拿着这份报表走进你老板的办公室时，你的老板已经忘记了他几个月前让你做的事情了。

再比较下面的例子。

你的老板走进你的办公室要一份数据分析，你在第二天就完成了这份分析。你第二天早上 8 点钟带着这份报表走进你老板的办公室，这个信息对你老板来说会有多重要？

关键就在于 6 个月后信息已经根本没什么用了，信息只有越新越及时，才能在商业中越有可能是有用的。

信息的时间价值是确实存在的，而且 DW2.0 环境可以大大加快信息访问的速度。

图 17-12 指出了信息的时间价值。

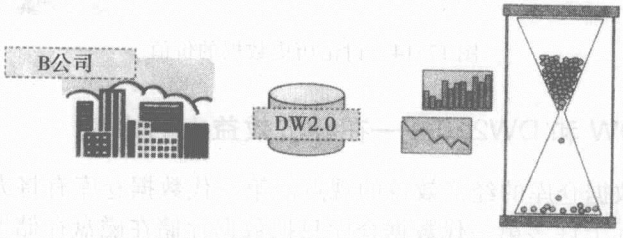


图 17-12 信息的时间价值又是多少

17.13 整合的价值

但是还有其他一些主要好处没有在原始的经济效果计算公式中体现出来。比如还有整合的价值。

对许多企业来说，仅是整合，DW2.0 环境的成本就是值得的。通过整合，来自一个

位置和一个应用的信息可以与另外不同的位置和应用的信息连接起来并相关联。

这种连通性的结果就是整合。而有了整合，就可以完成一些有价值的东西，例如用户可以对数据创建一个真正的360°的视角。

图17-13显示了整合是DW2.0环境的一种无形的利益。

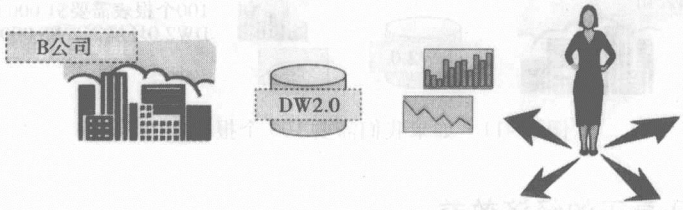


图17-13 整合信息的价值又是多少

17.14 历史信息

DW2.0环境还有一个无形的利益，就是从时间的角度来查看数据的能力。

在DW2.0出现之前有许多遗留系统。这些系统大部分都是面向事务的，因此哪里存在事务，哪里就需要提高性能。而当需要提高性能时，就需要移除系统中一些不必要的数据。系统中不必要的数据就好像人体中的胆固醇一样。

因此，旧遗留环境中的趋势是尽可能快地丢掉历史数据。系统中的历史数据越多，系统运行得就会越慢。最终结果就是在遗留系统环境中只有最少量的历史数据。

但问题是历史数据是有实际价值的。

如果遗留的事务处理环境不是存储历史数据的位置，那么DW2.0环境就是存储历史数据的位置。

图17-14显示了DW2.0环境是存储历史数据的位置。

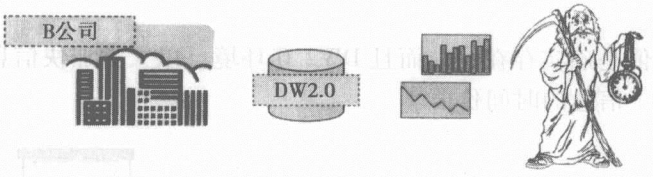


图17-14 讨论历史数据的价值

17.15 第一代DW和DW2.0——在经济效益上的比较

还有一种关于数据仓库的经济效益的观点。第一代数据仓库有将大量数据存储在磁盘存储上的习惯。实际上许多第一代数据仓库只把数据存储在磁盘存储器上。

但是DW2.0认为除了磁盘存储器外最少还要在两个地方存储数据。DW2.0认为，当数据还会被访问但是访问机率较低时应该存储在近线存储中，而且归档数通常也不存储在磁盘存储器上。

因为对于使用不同的存储设备，DW2.0的成本也明显地少于传统的第一代数据仓库的成本。图17-15比较了两代数据仓库的存储介质。

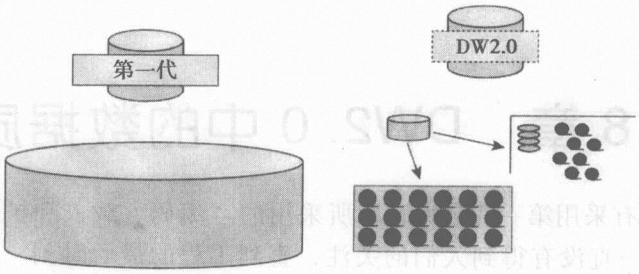


图 17-15 DW2.0 数据仓库的成本将比传统的第一代数据仓库大幅度地减少

17.16 企业用户的观点

成本分析是一种在 DW2.0 的生命周期内只需完成一次的活动。在所有正常的情况下都没有必要再做重复的分析。

企业用户会在很大程度上参与 DW2.0 成本分析的过程。实际上，企业用户在很多方面都会运用成本分析。

可以理解为企业用户直接或者间接投资 DW2.0 的基础结构，因此成本分析的结果必须使这些企业用户和他们的管理层感到满意。

17.17 总结

在宏观上对 DW2.0 进行成本分析是一件很难的事情，因为在许多宏观因素下，DW2.0 环境所带来的好处不能体现出来。相反，DW2.0 环境最好的成本分析方法是从微观上来看待成本分析。

看看这两个公司——一个拥有 DW2.0 环境而另一个没有，然后看看下面几项的成本：

- 查找数据。
- 收集数据。
- 整合数据。
- 分级数据。
- 汇报和分析数据等。
- 数据仓库的成本。

对很多报表和分析查看上述因素，DW2.0 环境的成本会比没有 DW2.0 环境的低很多。但是，还有其他一些非经济因素需要考虑：数据的时间价值，数据整合，以及现在已经可用的历史数据（第一次）。DW2.0 环境开启了一扇大门，使得组织能够去查看数据并做数据分析，这些都是它们以前做不到的。

第 18 章　DW2.0 中的数据质量

DW2.0 环境没有采用第一代数据仓库所采用的“编码、载入和扩展”标准。在这之后，数据质量问题一直没有得到人们的关注，直到工程的最后时刻。当项目组从源系统中提取数据并将其载入数据仓库时，会发现在源数据中存在着诡异的“精怪”。这使项目组成员非常困惑，因而不可避免地导致了预期计划的极大延迟。在测试和载入阶段发现的数据质量问题将成为项目不能按期完成并且超出预算的一个重要原因，而不论是哪种结果，都将导致项目指标无法达标。

下一代数据仓库方法 DW2.0 保证了数据质量小组进入数据仓库，就好像火车驶入了一块新的领土。在很早以前，火车上的士兵中有一些专门侦察火车将要驶过的地域情况的侦察兵，因而数据质量小组就是数据仓库开发团队的“侦察兵”。他们必须分析出交互区中的数据源，进而修复数据中存在的问题和异常，然后预先向开发团队发出警告。数据分析小组必须由企业人员和信息技术人员共同构成。

异常数据发现任务必须被编入方法中，然后在迭代计划开始之前运行该方法。（数据分析只是分析数据质量所必须完成的一项工作，另一个任务是检验数据是否遵循业务规则。）通过这种方法，项目组就能提前发现许多数据陷阱，从而做好充分的准备。

当发现数据质量问题时，必须马上上报给公司，公司必须决定哪些数据质量问题是它所关心和需要修复的。值得注意的是，并不是所有的数据质量问题对企业业务来说都是重要的。当某项业务已经表明哪些数据质量问题比较重要时，就需要确定用以清理数据的业务规则。这些规则产生了源数据转换的规格说明，从而避免了编码、载入和扩展的过程。

在下一代数据仓库 DW2.0 中，数据质量小组同时也被寄希望于能够从一系列的策略中选择出一些能够解决数据质量问题的策略。以下是其中的一部分策略：

- 修复源数据：实际上就是进入数据存储区，进而从物理上修复数据。
- 修复源程序：应用正确的编辑方法来验证数据。
- 修复业务过程：一个不完整的业务过程通常就是低数据质量的主要原因。
- 针对变化的调整：识别和解决这样的情况，即某个数据属性正在以其他一些目的被使用，而不是它的原始目的。例如，一个性别标记包含多于两个不同的值。
- 转换数据：转换数据以使其能够进入数据仓库——这是最常见的但不是唯一应该被采用的策略。

在理想世界中，人们希望能够找到问题的原因进而修复它，而不是仅仅修复所产生的结果。在源头修复数据和程序以及修复业务过程共同构成了对问题原因的修复。

必须指出的是，在数据进入整合区前有两种方法能够转换数据。第一种方法是简单的改变数据然后将它载入数据仓库。第二种方法也这么做，但是却载入了更多，因为它不仅会载入改变后的数据，而且还会载入改变前的数据。在很多情况下这可能是一个更好的策略：因为公司能看见你对数据的具体操作，所以他们更愿意相信这些数据。

18.1 DW2.0 中的数据质量工具集

其中当然包含了很多类型的工具。数据分析工具用来查找数据中存在的问题，还有另一组工具用来修复数据异常。DW2.0 中还有一些用来监视和汇报数据质量的工具。而负责选取、转换和载入数据的 ETL 工具通常也具有数据质量管理能力。

以下列出了四个在 DW2.0 的数据质量工具集中最基本也是最主要的部分（图 18-1）：

- 查找——分析并且查找数据异常。
- 修复——清除一些不遵循某些特定法则的数据。
- 移除——ETL 或者 ELT 工具能转换数据进而导入数据仓库。
- 监视和汇报——监视和汇报数据质量。

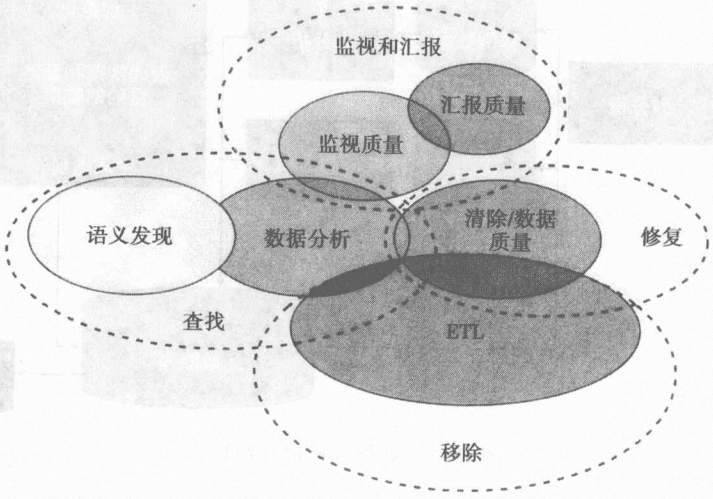


图 18-1 数据质量工具功能分类

在第一代数据仓库中，这四个部分表现为独特的专用工具类。尽管它们中的许多在 DW2.0 数据仓库中仍然拥有自己具体的功能域，但是它们的作用正在增强，如今已经在邻近域中存在与其他工具重叠的功能。一些工具甚至能分析半结构化数据工具的新的功能类别已经出现，例如，语义检查工具，它通过模糊逻辑发现数据中的规则。数据质量工具供应商之间的合并与收购已经导致了数据质量工具集功能的增强。在此背景下，DW2.0 的设计师们必须想出一个能够集查找数据质量异常、修复数据质量问题、移除数据和汇报/监视数据质量于一体的工具集。

18.2 数据分析工具和逆向工程数据模型

是否有可能手动地完成数据分析工作呢？答案是肯定的。以下就是一些可选择方案：

- 公司可以额外雇用一些用来梳理数据库数据的工作人员，让他们找出数据库中的重复记录并复制这些记录。但是通过这种方法却不能了解文件之间或者文件/系统之间的关系，并且这种方法开销也很大，因为新进的工作人员都必须经过培训和监督以确保他们能够遵循商业法则。
- 另一种方法是编写程序来发现数据异常。这种方法最典型的特点就是只能查找出

已经出现过的问题。有可能程序运行了很长一段时间，但是却没有产生任何效果。这种现象出现的原因就是“对于没有出现过的问题，我们一无所知”。

- 发现数据质量问题最好的方法是使用数据质量分析工具。

如今已经有很多数据分析工具可供数据质量小组使用（图18-2）。这些工具能使我们更容易地分析表中同一列的数据值，有时同时在一个表的不同列查找，有时在不同的表间查找，甚至是在不同系统的选定列中查找这些数据是否存在某种规范。这些规范能够使我们发现一些隐藏的商业法则，例如，每次当第一列中的值为“a”时，第五列中的值就为“x”或者“y”。

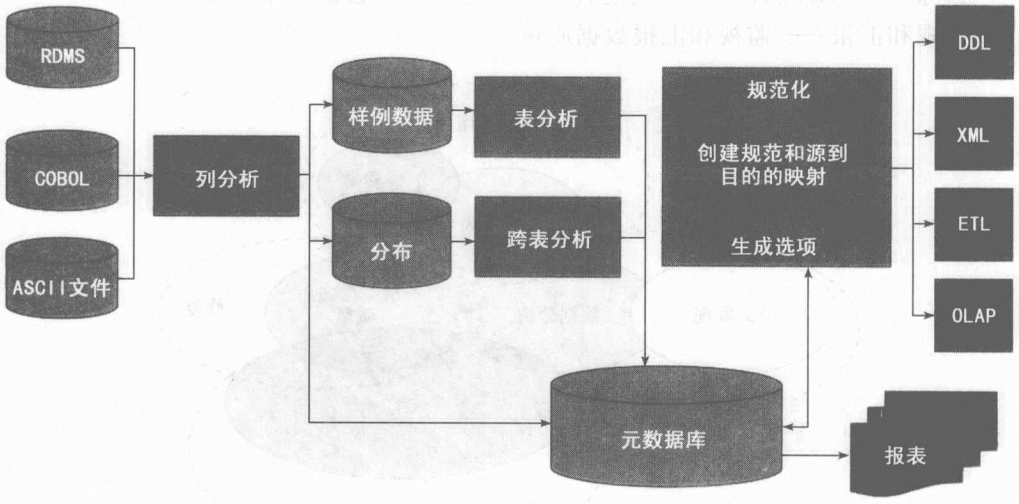


图18-2 数据分析过程组件

最好的数据分析工具比普通的工具更进一步。当分析完一个系统中一列的实际数据之后，好的数据分析工具会提出一个标准化模式。在自下而上分析的基础上，通过从物理数据库的实际数据值中提取出的数据可以建立一个第三范式数据模型。这一提取出来的数据模型是自上而下的数据建模过程中的一个重要输入，该建模过程必须和数据仓库并行运行。事实上，确保为DW2.0数据仓库开发高质量的数据模型构架非常有助于改进企业数据仓库的数据质量。

18.3 数据模型种类

那么在一个好的DW2.0构架下拥有哪些数据模型呢？在回答这个问题之前有必要做一些级别设置。图18-3的表格描述了在第6章介绍的Zachman框架数据列中的六行。

上三行是从技术独立的角度出发的，而下三行则是从技术依赖的角度出发的。第一列包含了一些从计划者的角度或者是企业的范围和规模出发得到的数据对象。第一行中的每一个数据对象都能够继续分解为主要的数据实体。第二行是从所有者的角度出发的，企业的主要概念都通过一个数据模型来表示，这个模型通常被称为实体关系模型或者实体关系表。第三行从设计者的角度出发，包含了逻辑数据模型。这是一个典型的完全第三范式数据模型。

技术依赖数据表示位于Zachman框架数据列的第三行和第四行的分隔线以下。第四

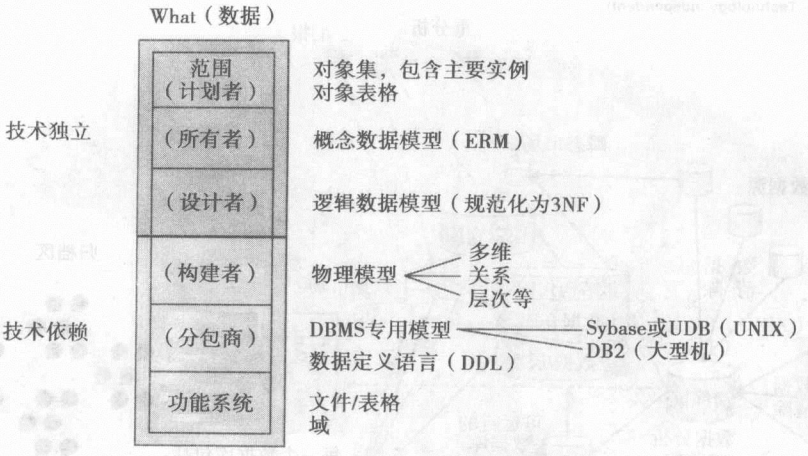


图 18-3 数据模型种类：Zachman 框架

行代表了物理数据模型。在一个数据仓库中可能存在多个物理数据模型，一个为主要的数据仓库服务，其他的为数据市场服务，还有一些物理数据模型则为数据仓库环境中的其他一些组件服务。这个级别的模型可以是一个包含关系模型、三维模型和非规范的物理数据模型的混合模型。必须指出的是，星型模式和雪花模式实际上就是物理数据模型。

数据定义语言（DDL）位于数据列的第五行。那些能够用来创建文件和表格的指令则位于第六行。

图 18-4 表示了各种用来充实 DW2.0 的组件。它们被叠加在 DW2.0 的四个主要区域——交互区、整合区、近线区和归档区上。每一个数据存储都必须能够追溯到环境级、概念级、逻辑级、物理级、构造级以及实例级。

一个好的概念数据模型能够突出企业的主要概念以及它们之间的联系。一个好的第三范式逻辑数据模型包含和展示了所有与业务实体、基数以及属性间的关系相关的属性。逻辑数据模型给出了一个很强大的逻辑视图，包含了企业以及企业的数据，因此，应该作为第三种模型类型——物理数据模型的起始点。

DW2.0 整合区的物理数据模型之间在结构上存在很大区别。它们有可能是适用于数据仓库枢纽的正常化和近正常化模型，也有可能是适用于数据市场的星型和雪花型模式模型。还有一些数据模型结构适用于探索数据仓库、数据挖掘仓库、可运行的数据库以及“oper marts”（可运行的数据集市）。那些需要被传递到近线区的数据必须离第三范式结构越近越好。在数据进入归档区时进行重构是一件再正常不过的事。

数据模型间多方位的溯源对 DW2.0 数据环境来说非常重要。它必须能够浏览来自一个从逻辑模型甚至是更远的概念模型备份的物理模型。同样地，它也必须能够自上而下地移动，从概念模型到逻辑数据模型，再到物理数据模型。一整套相互关联的模型从提高企业的数据质量、连接业务含义和结构化的业务规则，到形成数据实体和属性的物理实例需要经过一段漫长的过程。

那么究竟在 DW2.0 数据构架中哪个才是关键数据模型呢？图 18-5 所示的图表回答了这个问题，并且列出了所有在下一代数据仓库 DW2.0 中用到的关键数据模型。

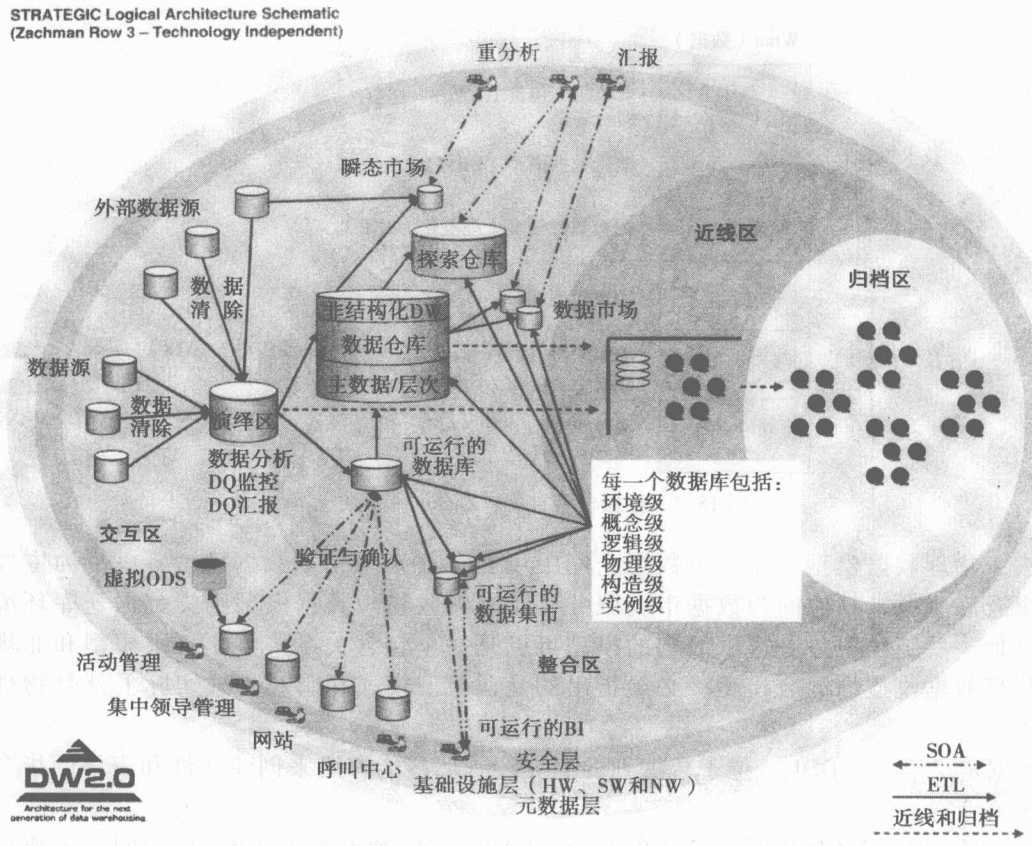


图 18-4 DW2.0 数据库一览图

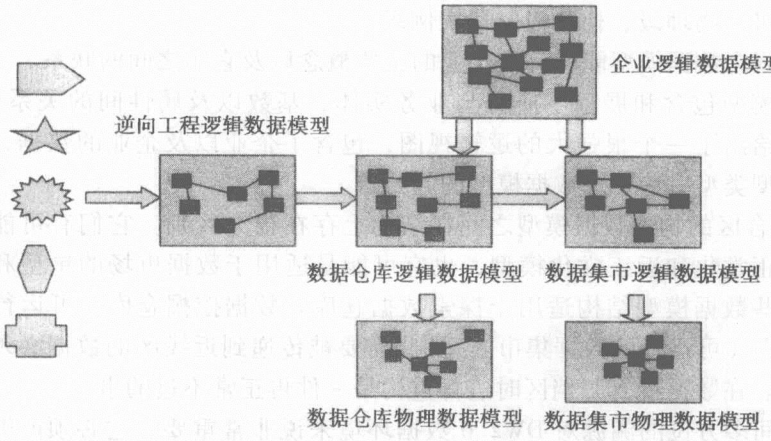


图 18-5 第二代数据仓库 DW2.0 中的数据模型种类

每一个源系统都是逆向创建逻辑模型的。通常一个企业的逻辑数据模型被用作数据仓库逻辑数据模型和数据市场逻辑数据模型的基础，它来自概念模型。同时适用于数据仓库和数据市场的物理数据模型则来自不同的逻辑数据模型。

接下来更深入地讲解企业的逻辑数据模型与其他逻辑数据模型之间的关系。例如，图 18-6 所示的数据仓库的逻辑数据模型和企业的逻辑数据模型之间的关系。

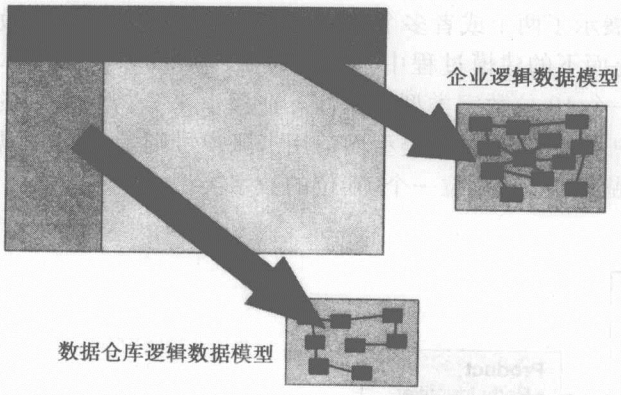


图 18-6 企业与数据仓库逻辑数据模型的比较

图表中的大矩形代表 Zachman 框架中第一列第三行的单元，即逻辑数据模型（LDM）单元。Zachman 框架中的每一个单元都有一定的“尺寸范围”和“详细尺寸”，它们分别用 LDM 单元顶部的薄横向矩形和 LDM 单元旁边的薄垂直矩形来表示。对企业逻辑数据模型中的数据创建一个企业范围的视图是一种值得提倡的方法。一个企业逻辑数据模型涵盖了企业的整个范围，但却并不那么详细。

一个模型，例如整个世界的所有属性往往会花费很多的时间和金钱。因为企业逻辑数据模型只描述了企业的主要属性和这些属性之间的关系。

数据仓库每一次连续的迭代都是逻辑数据模型驱使的，而逻辑数据模型的建立又要归因于迭代所需要的实例。拥有一个企业逻辑数据模型的好处就是可以不需损坏逻辑数据蓝图就允许该类渐进发展的发生。很多数据质量问题产生的原因都可以归咎于不适当的、不协调的逻辑数据建模。除了企业逻辑数据模型，找不到其他方法将问题数据集中起来；持续下去将导致完全的孤立，从而出现数据冗余和数据松散。

18.4 数据分析不一致对自上而下建模的挑战

在图 18-7 的例 1 中，通过数据分析发现的各种数据不一致问题都保存在一个文件中。在自上而下的数据建模过程中，一个叫做“Party”的实体被定义成一个个人或者单位。子类型“Individual”包含了一个属性“Gender”，其中 Gender 包含了男和女两个变量。同时，该团队正在使用数据分析工具对源系统中的实际数据进行分析。数据分析工具发现有七个值已经被 Gender 属性域使用过。这很快就能成为该域因为某种目的使用了该属性而不是单纯地反映个人的性别信息的证据。类似 Gender 属性的多目的使用以及它们隐含的实体都能够成为向逻辑数据模型添加相应实体和属性的基础。

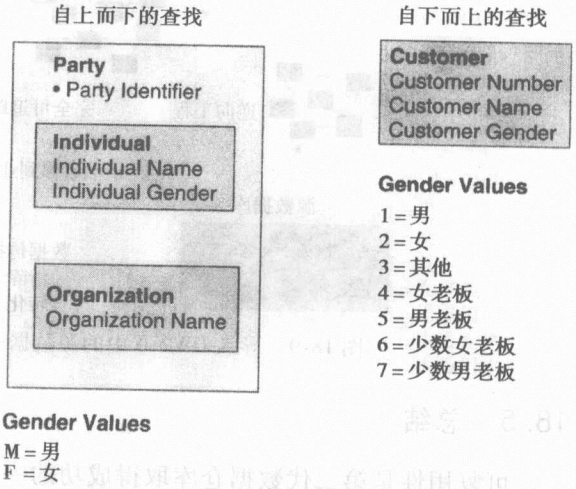


图 18-7 例 1：在一个文件内查找数据不一致问题

图 18-8 的例 2 展示了两个或者多个文件中的数据不一致是如何改变逻辑数据模型的创建方式的。在自上而下的建模过程中，一个叫做“Product”的实体应该被创建。产品序列号则被定义为一个 10 位数字类型的属性。在接下来分析一系列的源系统时，它能成为证明只有支票账户号是 10 位数字类型而信用卡账户号则是 16 位数据类型的证据。逻辑模型现在能够将产品序列号转化位一个 16 位的数字类型。

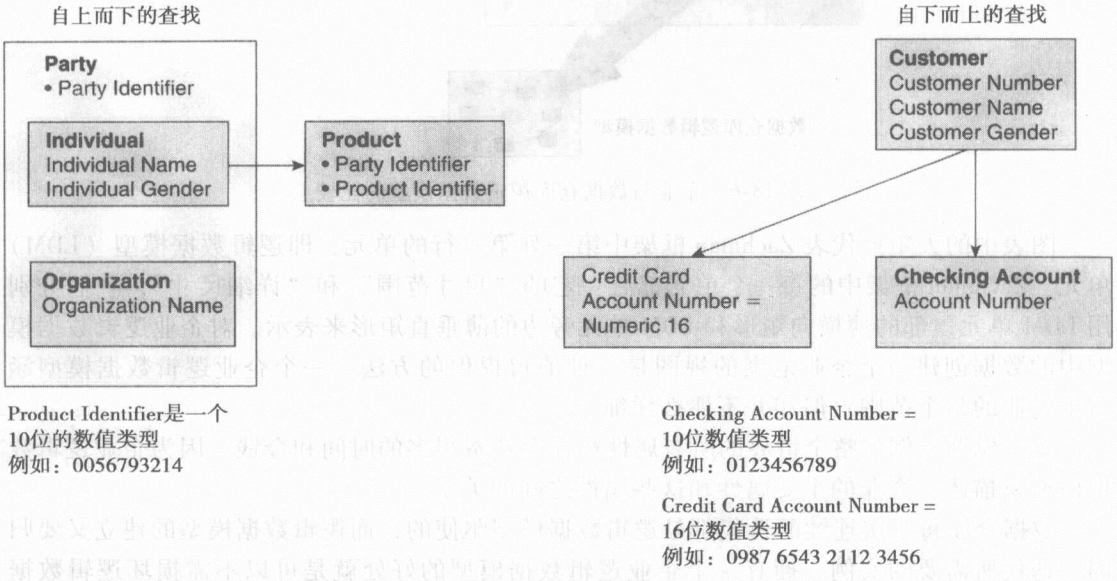


图 18-8 例 2：跨文件查找数据不一致问题

因此建议的最佳做法是将受数据分析工具控制的自下而上的数据建模与严格的自上而下的数据建模过程结合，这样才能为下一代数据仓库构造出最好的数据构架。一个类似图 18-9 中描绘的坚实的数据构架是提高数据质量的关键因素。

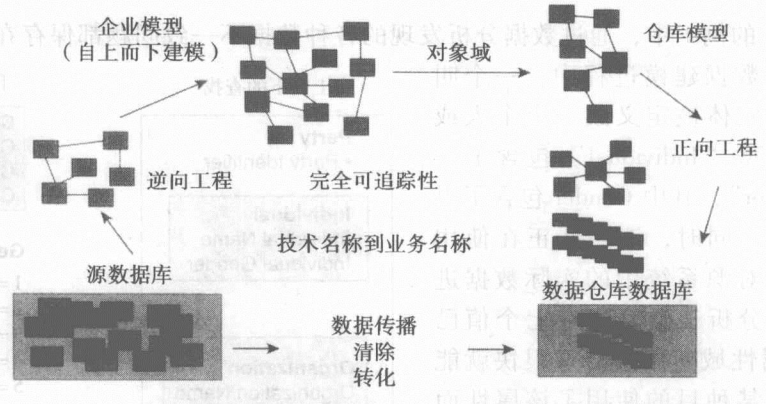


图 18-9 导入 DW2.0 中的源数据——一个坚实的数据构架的组成

18.5 总结

可复用性是第二代数据仓库取得成功的一个关键因素，而支撑着整个程序的数据模型的质量问题应该得到更多的关注。所构造的模型必须准确地反映业务，并且能够在程序

的升级版本中重复使用。

最基础的模型是企业数据模型。在传统方法中，这种模型是使用自上而下的方式，通过利用共同的需求计划和应用设计技术导出的。这些技术能够很快得到一个很好的模型比较。但是缺点也同样明显，它只建立在管理人员和高级分析师所分析出的商业法则上。在实际操作中，使用数据的系统可能拥有一些不同规则的集合。这是因为这些系统通常已经使用了20年了（甚至更久），当初对数据所作的改变未被保存，而大多数情况下，当初改变数据的人都已经不在公司任职了。

揭露数据真实情况的唯一方法是逆向将数据导入一个抽象的逻辑数据模型。第一代数据仓库在过去曾经尝试实现过，但是当时可用的工具有限。现在，一组新的工具集已经出现——数据分析工具。这些工具对于逆向控制数据以及自下而上构造数据模型来说是一个理想的帮手。当以这种方式构造一个模型时，它完全是建立在实际数据内容的基础上，因而发生错误的概率以及在数据建模过程中的遗漏都能减少。这种自下而上的模型被当作自上而下构造模型的输入；在效果上，自下而上模型用以挑战由企业草拟的自上而下模型。

企业数据模型并不是在一个成功的商业智能实施中所必需的唯一的数据模型。较好的建议是将 Zachman 框架的数据列中定义的所有模型都构造出来。

第 19 章　DW2.0 和非结构化数据

据估计，在企业里有 80% 的数据是非结构化文本。但不幸的是，当前计算机上运行的技术都是致力于处理结构化、可重复的数据的。这就导致在企业中做决策时没有利用到一些有价值的信息，文本中的有用信息没有成为决策过程中一个重要部分。

19.1　DW2.0 和非结构化数据

致力于下一代数据仓库的 DW2.0 构架意识到在非结构化的文本信息中存在有价值的信息，必须对文本做一些工作以使它适合分析处理。

出发点就是文件本身。图 19-1 显示了各种形式的文本，如电子邮件、文档、医疗记录、协议、电子表格以及声音副本等。

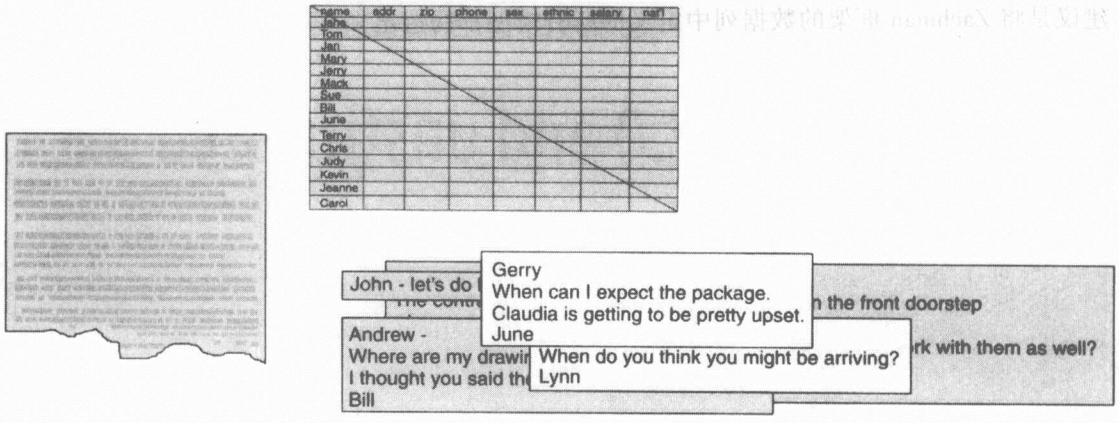


图 19-1　文档、电子表格、电子邮件——常见的非结构化数据类型

19.2　文本读取

为分析处理而准备非结构化数据的过程的第一步就是读入文本。文本存在于多种格式中，这些格式也可能需要读入。

图 19-2 描述了非结构化源文本的读入。

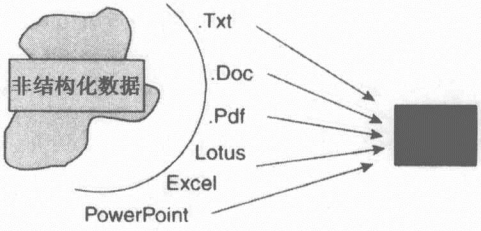


图 19-2　第一个任务是读非结构化数据

当原始的源文本被读入以后，下一步就是要准备这些数据以输入数据库。文本的准备是一个复杂的处理过程。有一些很好的理由表明文本必须被处理：

- 非结构化数据需要与关系型格式相匹配。
- 非结构化数据必须被“整合”，这样分析处理才有意义。如果仅将原始文本简单

地强制输入数据库，就会导致文本不能被有效甚至有意义地分析。

19.3 在哪里进行文本分析处理

现在即将做一个重要的战略决策，就是在什么位置进行文本分析处理。主要有两个可选择的位置，第一个就是非结构化数据所处的位置——非结构化环境，另一个就是结构化环境。在结构化环境中进行文本分析要求非结构化文本被读入、整合、处理及存储在结构化环境中。

毫无疑问，读入、整合、处理非结构化文本数据是一个艰巨的任务。但是，当非结构化文本经过处理并存储在结构化环境中时，就会出现很多机会。当非结构化数据被整合并存储在结构化环境中时，就可以使用标准的分析技术。

一些组织机构已经花了数百万美元来培训员工和用户，目的就是在结构化技术的基础上创建一个分析环境。结构化环境中已经有了数据库技术、智能商务、ETL、统计性处理等，利用这些已经存在的分析环境是非常有意义的。现在所需的就是读取与整合本文信息的能力，文本 ETL 的出现就是为了实现这一目的。

所以，选择一个环境来完成文本分析处理是比较容易的。结构化环境就是完成分析处理最好的地方。

19.4 文本整合

“整合”文本的过程要在将文本存储在数据库中之前完成，该过程有很多不同的方面。为了将文本存入数据库并作为 DW2.0 数据仓库的一部分，对其进行后续分析所需的准备文本过程包含了多个步骤。最重要的有：

- 简单编辑
- 移除无用词
- 同义词替换或串联
- 同形异义解析
- 主题性聚集
- 外部术语表/分类覆盖
- 分词
- 替换拼写解析
- 外语自适应
- 直接或间接搜索帮助

接下来是对这些非结构化数据各个方面的描述。

19.5 简单编辑

为分析处理准备非结构化文本的第一步是对格式、标点和字体等做一些简单的编辑工作。这种简单编辑是非常重要的，因为将来的分析性搜索不需要被印刷版式的差异所阻碍。例如，如果要对“bill inmon”进行搜索，则实际搜索还需要找出“Bill Inmon”，尽管这两个单词的首字母是大写的。图 19-3 描述了为分析处理而从非结构化文本中消除格式、字体和标点的情况。

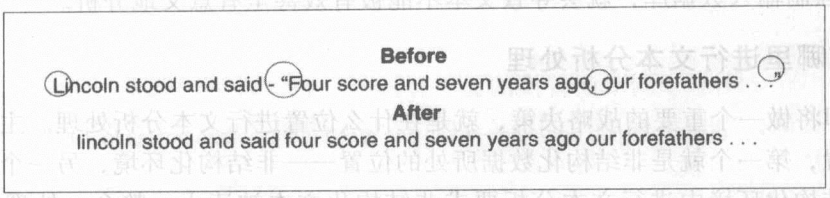


图 19-3 基本的标点符号、大写、格式、字体和其他被视为搜索障碍等方面的移除

19.6 无用词

下一步要做的就是消除无用词。无用词是一个有助于语言平滑流畅的词，但其本身却不包含什么信息和意义。一些典型的无用词如下：

- 一个（a）
- 和（and）
- 那（the）
- 是（was）
- 那个（that）
- 哪个（which）
- 到（to）
- 从（from）

图 19-4 描述了无用词的消除。

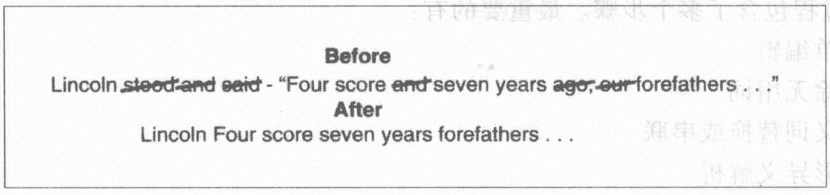


图 19-4 删除无用的单词

19.7 同义词替换

另一个在文本整合过程中可选择的步骤是同义词替换。同义词替换用来合理化使用不同术语的文本，使其都使用单一的术语。同义词替换使用一个标准用词来替换其他所有和其有相同含义的词。前后一致地使用同一种术语是保证可靠地、可重复地查询数据仓库中的非结构化数据的过程中的重要一步。图 19-5 解释了同义词替换。

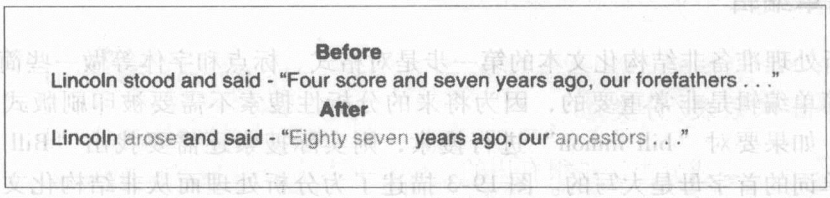


图 19-5 同义词替换

19.8 同义词串联

同义词串联是相对于同义词替换的另一种选择。在同义词串联中，不是用一个标准用词来替换同义词，而是将标准用词插入到所有出现的同义词后面或跟它们串联起来。图19-6解释了同义词串联。

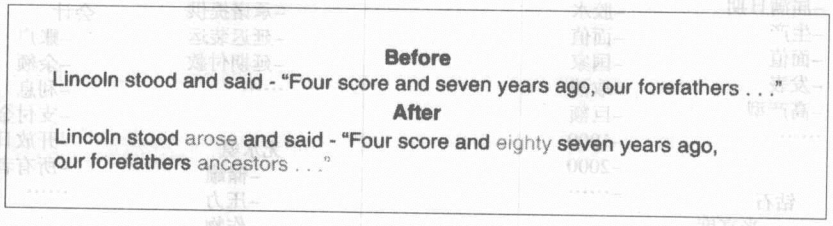


图19-6 同义词串联

19.9 同形异义解析

同形异义解析跟同义词串联和同义词替换正好相反。同形异义解析用来澄清那些有多种含义的单词或短语，用这些单词实际表示的意思来替换或覆盖出现在文本中的单词或短语。图19-7解释了同形异义解析。

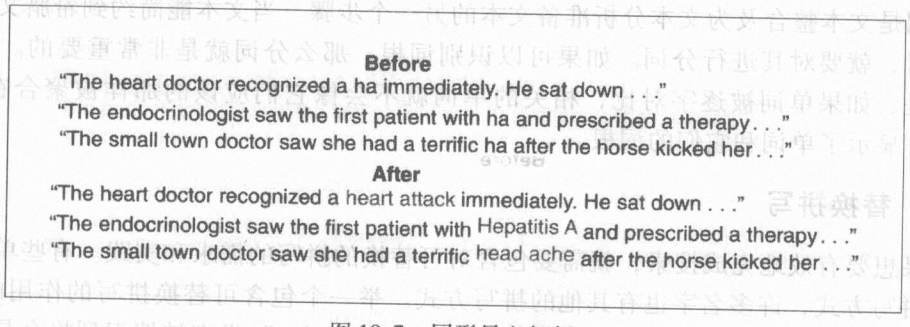


图19-7 同形异义解析

19.10 建立主题

文本整合后需要做的一个有趣且有用的针对文本的事情是产生一个文本的“聚类”，而聚类文本则生成“主题”。在文本聚类中，单词和短语根据它们出现的次数和彼此间的形似度而从逻辑上被聚合在一起。

聚类同样也能产生一个术语表或分类法。这个术语表和分类法被称为“内部术语表”或“内部分类法”，因为它是从系统内部的文本产生的。图19-8显示了文本聚类和生成的主题。

19.11 外部术语表/分类法

虽然内部术语表或分类法是很有用的，但外部术语表和分类法也同样很有用。外部术语表和分类法可以来自任何地方，如书籍、索引、网络等。外部术语表和分类法可表示任何事情，能用于文本上添加一个结构。文本可被读入系统，然后可作一个比较来确定

该文本是否属于或与外部术语表和分类法相关。

图 19-9 显示了一些外部术语表和分类法。

图 19-8 一些主题和它们的描述符

图 19-9 一些外部分类法

19.12 分词

分词是文本整合及为文本分析准备文本的另一个步骤。当文本能简约到希腊文或拉丁文词根时，就要对其进行分词。如果可以识别词根，那么分词就是非常重要的。换一种说法就是，如果单词被逐字对比，相关的单词就不会像它们应该的那样被聚合在一起。图 19-10 显示了单词和它们的词根。

19.13 替换拼写

如果想要有效地完成搜索，就需要包含对可替换的拼写的需求和实践。有些单词有可替换的拼写方式，许多名字也有其他的拼写方式。举一个包含可替换拼写的作用的例子，假设正在搜索“Osama Bin Laden”，如果“Usama Bin Laden”没有被搜索到将会是一件很遗憾的事，因为这个名字有不同的拼写方式。图 19-11 解释了为什么要识别出同一事物的不同拼写方式。

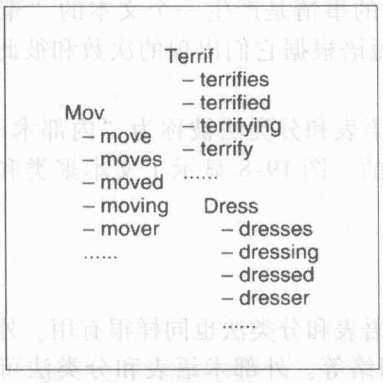


图 19-10 对单词进行分词

图 19-11 相同名字或单词的不同拼法

19.14 跨语言的文本

文本分析另一种有用的特点就是运用多种语言的能力。图 19-12 显示了不同的语言。

Este es un pobrecito
The girl ran inside her house
J'ai un chat, seulemente. . . .

图 19-12 不同语言的运用

19.15 直接搜索

文本分析还有另一种重要的特征，就是支持不同种类搜索的能力。文本整合需要为这种特征做好准备，其中一种需要支持的搜索就是直接搜索。直接搜索的典型代表就是 Yahoo 或 Google。将参数传给搜索引擎，然后该引擎查找所有出现搜索参数的情况。图 19-13 描述了直接搜索。

19.16 间接搜索

另一种搜索类型是间接搜索。在间接搜索中，搜索参数同样传给了搜索引擎，但却并没有对其进行搜索。相反，间接搜索是搜索任何与该参数相关的东西。例如，图 19-14 描述的对“Sarbanes Oxley”的间接搜索。该搜索并没有去查找“Sarbanes Oxley”，而是去查找与 Sarbanes Oxley 相关的文本。

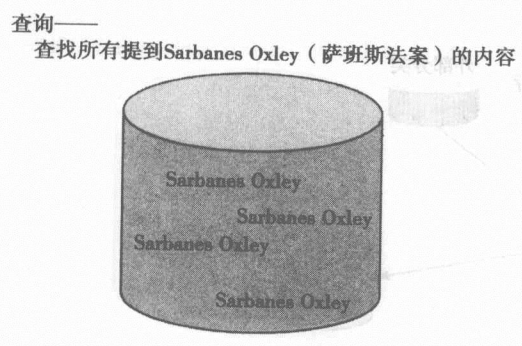


图 19-13 直接搜索

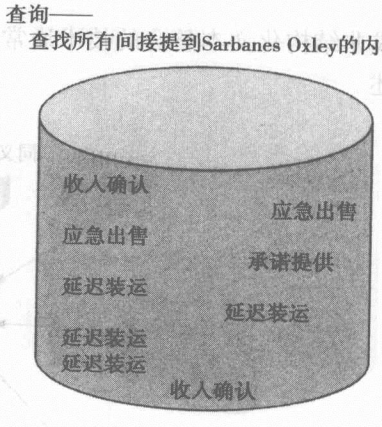


图 19-14 间接搜索

19.17 术语

在以分析处理为目的的文本处理过程中有个很大的问题，即术语的处理。术语之所以是一个问题，是因为语言常以术语的形式表达。设想人的身体。对人类身体的任何一个部分都有多达 20 种方式可以指出它。一个医生用这一套术语，另一个医生用那一套术语，而护士又用另一套术语。这些不同的人都在谈论同一件事，然而却用不同的语言。

如果想要对文本进行分析处理，就必须有对术语问题的解决方法。最终的单词和短语的文本数据库必须同时有一般性和具体性的存储。用于文本分析的最终文本数据库必须要有原始的医生或护士用过的具体的单词，也要有整个分析团体都能理解的一般术语。

如果一个组织不解决术语的问题，那就不可能完成有效的文本分析处理。

19.18 半结构化数据/值=名称数据

非结构化数据有不同的种类。最简单的形式就是文档中的文本。在文档的文本中，单词和短语都是没有顺序和结构的，一个非结构化文档仅仅是个非结构化文档。

然而，文本文档还有其他的形式。在某些情况下，文档的作者会给出一个可以推断出来的文档结构。一个简单的例子是烹饪书，在一本烹饪书中有很多烹饪方法。这是一个文档，里面有隐含的开始和结束。一个烹饪法结束就是另一个的开始。

很多时候，有必要将书中隐含的结构映射到文本分析数据库上。在某些情况下，这是一个简单且显而易见的事。在另一些情况下，如何映射却一点都不明显。

另一种在DW2.0环境下需要特殊处理的非结构化数据形式是一种被称为“值=名字”的数据形式。要理解这种类型的数据，试想一堆个人简历。在每份个人简历上都能找到公共的信息，如名字、地址、教育背景、工资等。能够理解在非结构化数据中哪种数据正被考虑是很重要的。换句话说，对于“名字—Bill Inmon”，系统能很方便地分辨出“名字”是一个很重要的域，且“Bill Inmon”是那个被命名的人。这种能力意味着文本能被读入，单词能按符号而不是按字面找出。这种能按符号感知单词的能力对建立文本分析数据库是很重要的。

19.19 准备数据所需的技术

完成非结构化文本整合的技术通常被称为文本ETL技术，图19-15对其在较高层次进行了描述。

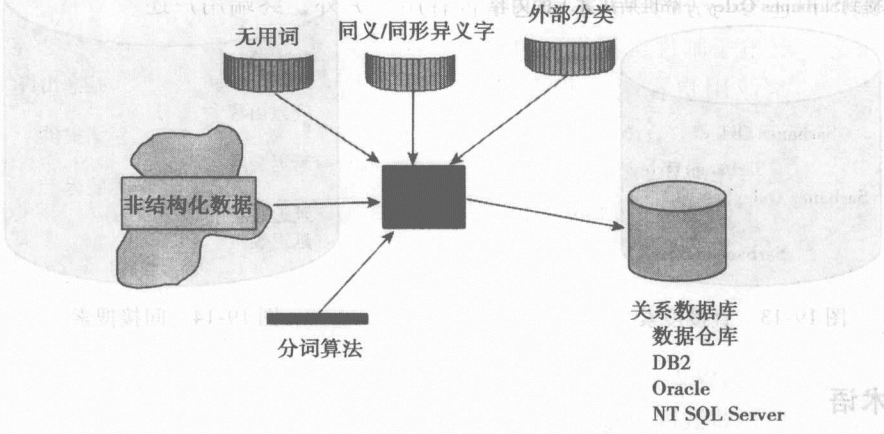


图19-15 非结构化数据怎样转化成关系数据库

19.20 关系数据库

当非结构化文本已准备好进行分析处理时，该文本被置于一个关系数据库中。该关系数据库可能会被各种不同工具访问和分析，比如商业智能工具。图19-16显示了在非结构化数据上的智能商务的利用。

19.21 结构化/非结构化连接

当非结构化关系数据库建立以后，它将被连接到结构化数据库，从而形成组织机构的DW2.0基础。

图19-17描述了两种不同类型的数据库之间连接的建立过程。

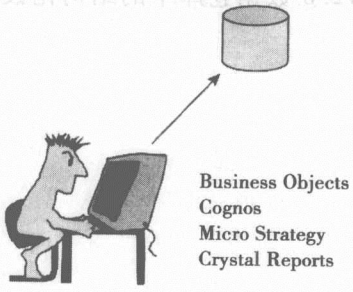


图19-16 一旦数据处于关系型格式，它便几乎可以被任意的商业智能工具访问和分析

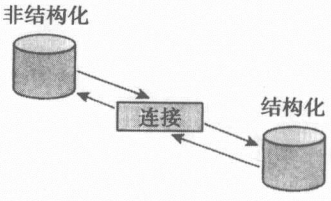


图19-17 建立结构化数据和非结构化数据之间的连接

19.22 企业用户的观点

如果说有与企业用户最接近的数据，那就是非结构化文本数据了。非结构化文本数据构成了终端用户的日常商务生活。所以当非结构化文本数据要被归入DW2.0环境时，终端用户需要高度参与。

企业用户参与无用词的具体说明。他们还参与确定术语以及选择在DW2.0中用哪种语言。终端用户还参与分词——不管这是否有用。另外，终端用户还参与确定非结构化文本数据的来源，如电子邮件、报告、协议等。

简而言之，企业用户在很大程度上参与了非结构化文本数据的获取、准备、输入DW2.0环境等各方面。

通常，企业用户在DW2.0的结构化部分的设计中只是被动地参与。但对DW2.0的非结构化文本部分却正好相反。例如，企业用户很大程度上参与了文本ETL转换的详细规格说明。

19.23 总结

非结构化数据是DW2.0数据仓库的一个重要组成部分。

非结构化数据必须被读入和整合到DW2.0环境中。非结构化数据的整合过程包括但不仅限于以下内容：

- 移除标点、字体等阻碍数据访问和分析的东西。
- 管理可相互替代的拼写。
- 分词。
- 无用词管理。
- 内部主题和分类法的建立。
- 同义词替换。

- 同义词串联。
- 同形异义解析。
- 外部分类/术语表分类。

当聚集和整合文本化数据以后，便建立关系数据库以支持分析处理。整合以后，文本数据以关系型格式放置，并建立一个关系数据库。然后这个关系数据库就可以进行商业智能处理。最后，这个非结构化关系数据库要和在 DW2.0 数据仓库中的结构化数据库进行连接。

第 20 章　DW2.0 与记录系统

DW2.0 数据仓库中的大部分数据集的建立都是以操作型或旧的应用系统为基础的。图 20-1 展示了这类数据源。

写于许多年前并且很多情况下无文档记录，最初的数据就是从这样的操作应用环境中进入企业环境，并且数据常以事务执行的副产品的形式进入交互区。

图 20-2 展示了旧操作环境中的典型要素。

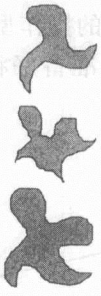


图 20-1　操作型/旧的系统环境

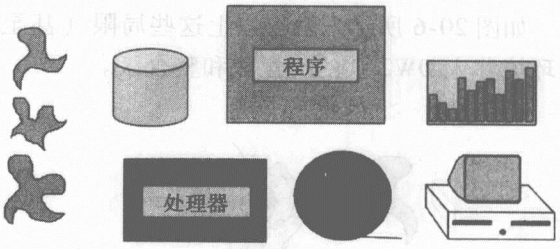


图 20-2　操作型环境中的元素

旧操作环境中的要素种类有程序、报表、处理器、文件和数据库。

由于进入数据仓库中的大部分数据是在操作环境中产生的，所以要对它格外关注。最终获取的数据要尽可能的准确、及时、完整，故需定义“记录源”数据系统，经确认的记录源系统是最佳的数据源。

为 DW2.0 寻找最佳的数据源与数据质量之间有非常重要的联系。为了得到好的数据质量，人们所进行的最重要的一步是谨慎地选择记录系统。也就是说，如果根本没有选择或是选择不够正确的话，就会反映出较差的数据质量。

寻找最佳数据源应从旧操作环境中的应用开始。图 20-3 描绘了那些应用。

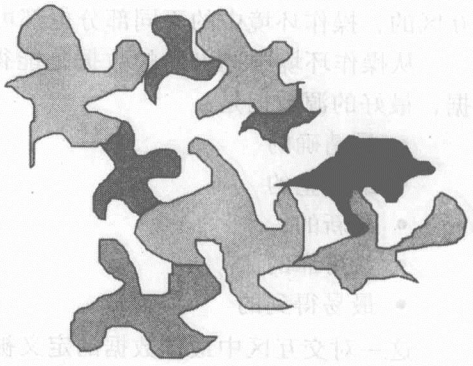


图 20-3　旧环境是一堆未整合的应用

许多企业的操作型应用系统环境受其内部发生的事务处理的约束。当企业的操作型事务处理发生时，尤其是有大量事务处理的时候，操作环境是不能受干扰的。在这种状况下，操作环境可能被认为是娇气的，在业务周期高峰期无法执行大量的批量处理。问题是许多环境中，操作处理过程窗口都很大，往往要持续很长时间，远远超过上午 8 点至下午 5 点这一时间段。

图 20-4 显示了在有些时刻，旧环境中的交易处理会导致操作环境无法为其他任何请求提供服务。

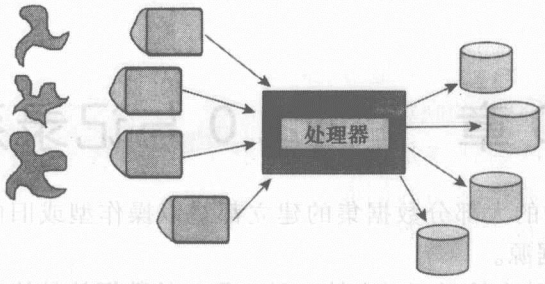


图 20-4 操作型环境通常受联机事务处理（OLTP）所需的规则的约束

还有一些其他的与旧操作环境相关的局限。其中之一是，很多情况下，建立起来旧的操作环境后没有相应的文档，或是没有最新的文档。而再返回去查找该操作型应用是干什么用的可不是件容易的事。图 20-5 说明存在着很多缺少文档的旧的操作型应用。

如图 20-6 所示，除去以上这些局限（甚至更多），构架师必须准备好将数据从旧操作环境移入 DW2.0 的交互区和整合区。

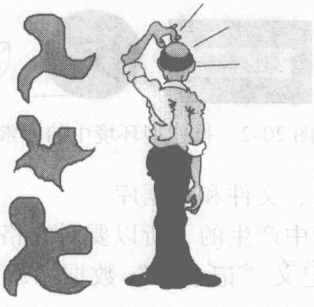


图 20-5 常常找不到说明文档

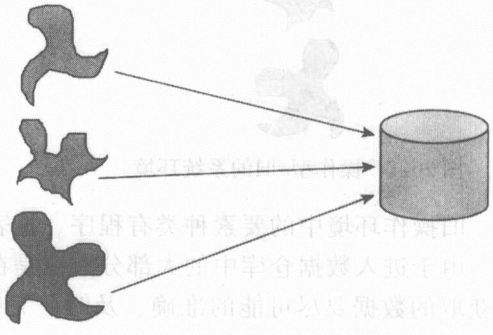


图 20-6 为数据向 DW2.0 移动作准备

数据仓库构架师的工作是找出旧系统中什么样的数据是数据仓库的最佳数据源。图 20-7 描述了这一任务。

不是所有操作环境中的数据都能进入 DW2.0 的交互区的，操作环境中的不同部分也都可当作数据源。

从操作环境中选出来的数据是能得到的最好的源数据，最好的源数据是：

- 最精确的
- 最完整的
- 最新的
- 最可靠的
- 最易得到的

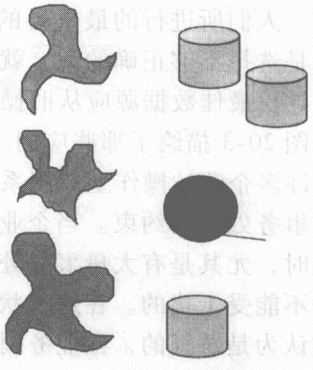


图 20-7 寻找最佳数据源

这一对交互区中最佳数据的定义被称作源数据记录系统。当数据从整合区进入归档区时，记录系统会有所延伸。

有时，两三个文件会被用作 DW2.0 交互区中同一数据单元的数据源，在一些其他情况下，操作型应用系统环境中只有一个单独的数据源。

在对来自于操作环境的最佳源数据做了明确的定义后，就需要在源数据到目标数据之间建立映射关系，这一过程称为数据映射（图 20-8）。有些情况下，映射关系简单到只说明数据从一个地方开始到另一个地方结束。图 20-9 描绘了一个简单的映射过程。

但在其他情况下，映射往往是更复杂的。图 20-10 表明，在数据移动的同时可能还需要进行计算，不仅需要计算，还需要知道计算的日期和速率。

简单的计算可能不是唯一必须的计算类型。图 20-11 给出了一种不同的计算，在这种计算中用到了多个不同的记录。计算往往并不是复杂的，但其中的数据安排却是非常有挑战性的。

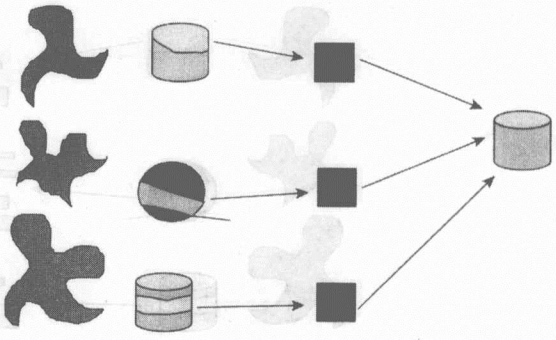


图 20-8　建成的由旧环境向数据仓库的映射

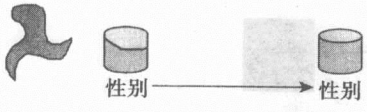


图 20-9　一个简单的映射

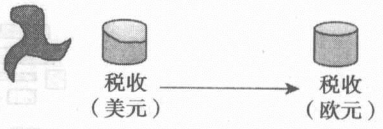


图 20-10　一个较复杂的映射

当有不止一个数据源时，会产生另一种形式的映射。这时，还需要用来确定哪种数据源在哪种条件下最佳的逻辑。图 20-12 描述了这种逻辑。

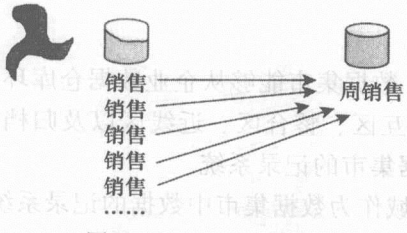


图 20-11　另一种映射

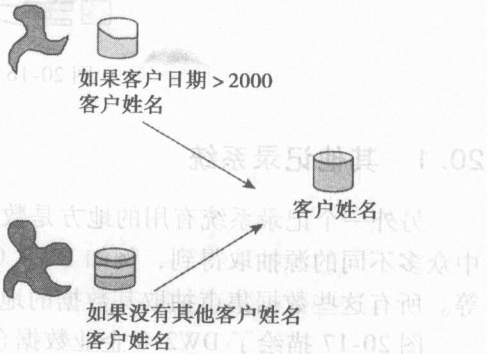


图 20-12　有多个源时，使用逻辑来判定哪个源最佳

在一些情况下，找不到数据源时就需要提供一个默认值（图 20-13）。

数据映射的另一考虑是如何协调不同的编码值。有时，源数据采用一种编码方式，而目标数据需要采用不用的编码方式。图 20-14 描绘了编码值间的协调。

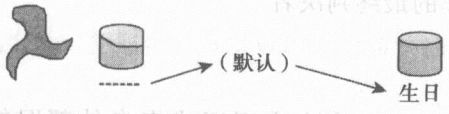


图 20-13　有时需要提供默认值

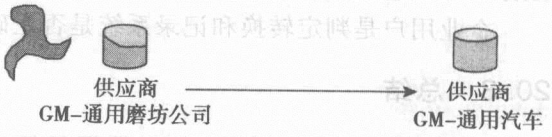


图 20-14　映射包含校正编码值

建立好映射之后，交互区的数据增长就可以开始了。图 20-15 展示了从源到目标的整个数据映射。

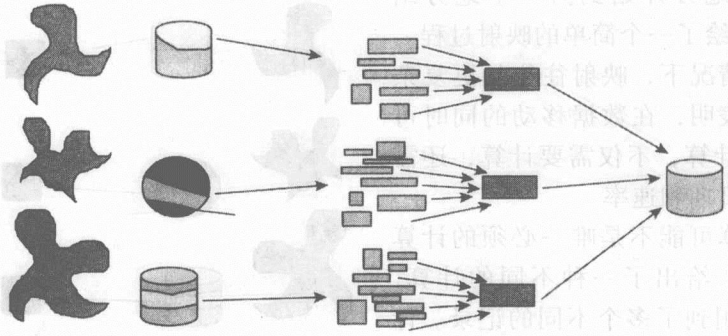


图 20-15 映射完成后，DW2.0 的装入过程就准备开始了

数据映射是 ETL 过程的必要输入（图 20-16）。

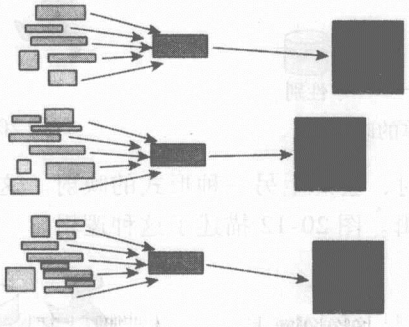


图 20-16 映射说明变成 ETL 说明

20.1 其他记录系统

另外一个记录系统有用的地方是数据集市的创建。数据集市能够从企业数据仓库环境中众多不同的源抽取得到，例如 DW2.0 数据仓库的交互区、整合区、近线区以及归档区等。所有这些数据集市抽取其数据的地方被称作对数据集市的记录系统。

图 20-17 描绘了 DW2.0 企业数据仓库中不同的区域作为数据集市中数据的记录系统。

20.2 企业用户的观点

从重要性角度来说，企业用户能够做的有助于 DW2.0 构建的最重要的一件事是对记录系统的详细规格说明。业务是证明记录系统的最终权威，如果这个系统不能反映企业业务，那么它就是没有意义的。

企业用户是判定转换和记录系统是否正确工作的最终判决者。

20.3 总结

数据仓库环境包含数据源和数据目标。数据源——来自交互区或来自外部旧的应用——被称作记录系统，记录系统是对最佳数据源的定义。

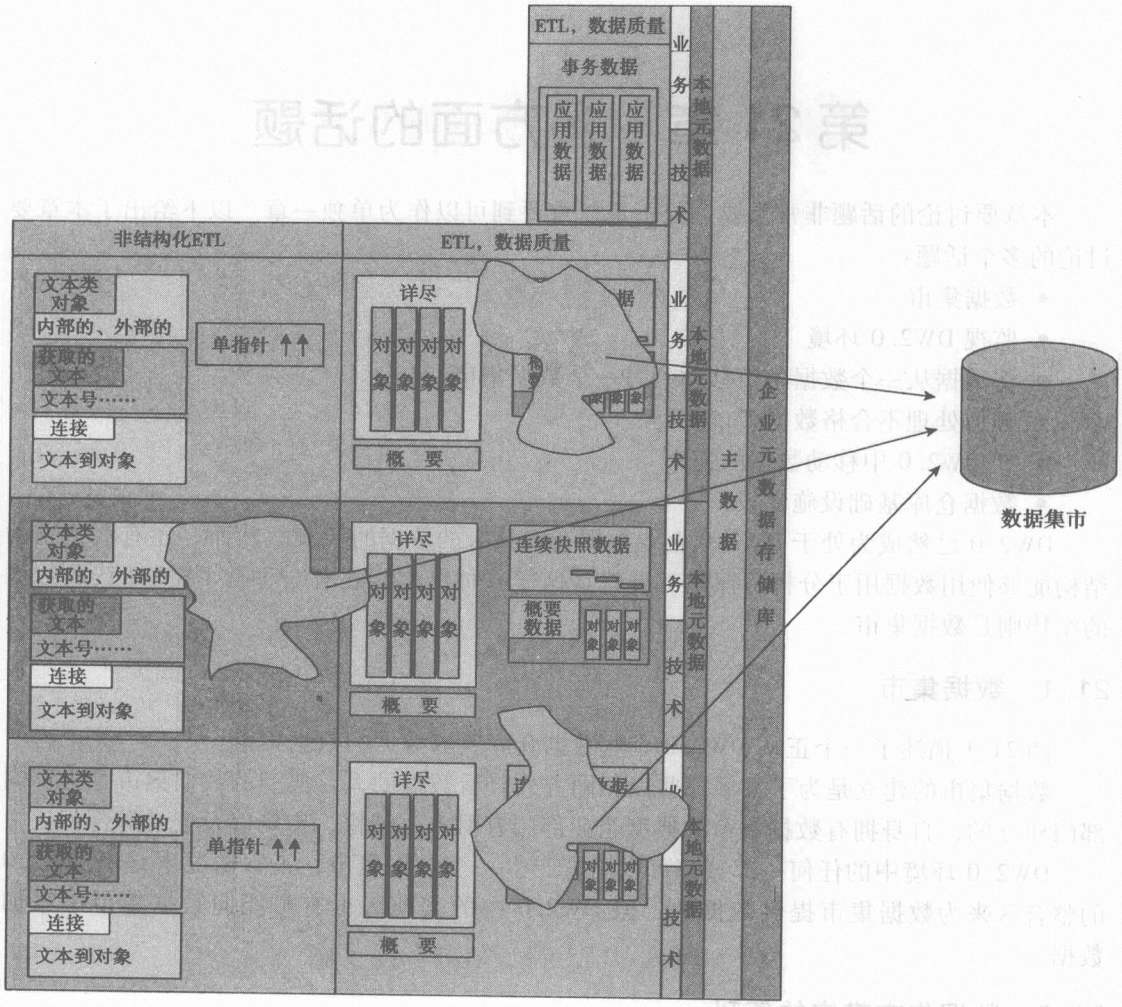


图 20-17 数据集市同样也有记录系统

最佳数据源是那些完整、准确、实时的数据。高质量的数据能够使记录系统更好。

记录系统的重要性有多方面原因，它对于想要将记录系统产生的映射用于提供目标数据的开发人员而言很重要，更重要的是，终端用户分析师需要将记录系统作为分析过程的一部分加以理解。

记录系统是环境中数据整合的一个主要贡献者。

第21章　多方面的话题

本章要讨论的话题非常重要，但不足以重要到可以作为单独一章。以下给出了本章要讨论的多个话题：

- 数据集市
- 监视 DW2.0 环境
- 将数据从一个数据集市移动到另一个数据集市
- 如何处理不合格数据
- 在 DW2.0 中移动数据的速率
- 数据仓库基础设施建设

DW2.0 已然成为处于 DW2.0 企业数据仓库核心的数据的代表。然而，还有些独立的结构能够使用数据用于分析，探索工具就是这样一种结构。还有一种从 DW2.0 获取数据的结构则是数据集市。

21.1　数据集市

图 21-1 描述了一个正从 DW2.0 企业数据仓库获取数据的数据集市。

数据集市的建立是为了方便那些以相同方式查看数据的人。典型的数据集市是为不同部门建立的，自身拥有数据集市的典型企业部门有财务、销售、市场以及会计等部门。

DW2.0 环境中的任何一部分都可以用来产生一个数据集市，正常情况下是由 DW2.0 的整合区来为数据集市提供数据的，但 DW2.0 中的其他区也有可能向数据集市中添加数据。

21.2　数据集市带来的便利

数据集市带来的便利就是，DW2.0 中的数据是过于详细的数据，而数据集市中的数据通常不是那么过于详细。当人们以共同的方式来查看数据时，更有效也更方便的做法是，获取详细数据并按照用户组想要查看它的方式将其结构化。这样，当某人想要查看数据时，总能以个人想要的结构化、格式化的方式查看数据，而不必担心还需提取详细数据并将其重构的工作。

数据集市如此盛行还有其他一些重要原因。它之所以吸引人是因为当将数据置于企业数据仓库外部时，数据的处理成本通常会降低，在 DW2.0 的宿主机器上的处理成本常常与计算周期能达到的最高值一样高，而在脱机状态下提取数据再将其放入另一个更小、更部门级的机器上时处理成本就会减小。

数据集市盛行的另一个原因是通过将数据集市提取到另一机器上，DW2.0 企业数据仓库环境的机器周期得以保留，而将机器周期从 DW2.0 环境移动到另一个环境极大地提升了主 DW2.0 的性能。

将数据集市提取到另一机器上是个不错的主意的另一个原因是，不同部门喜欢这种对

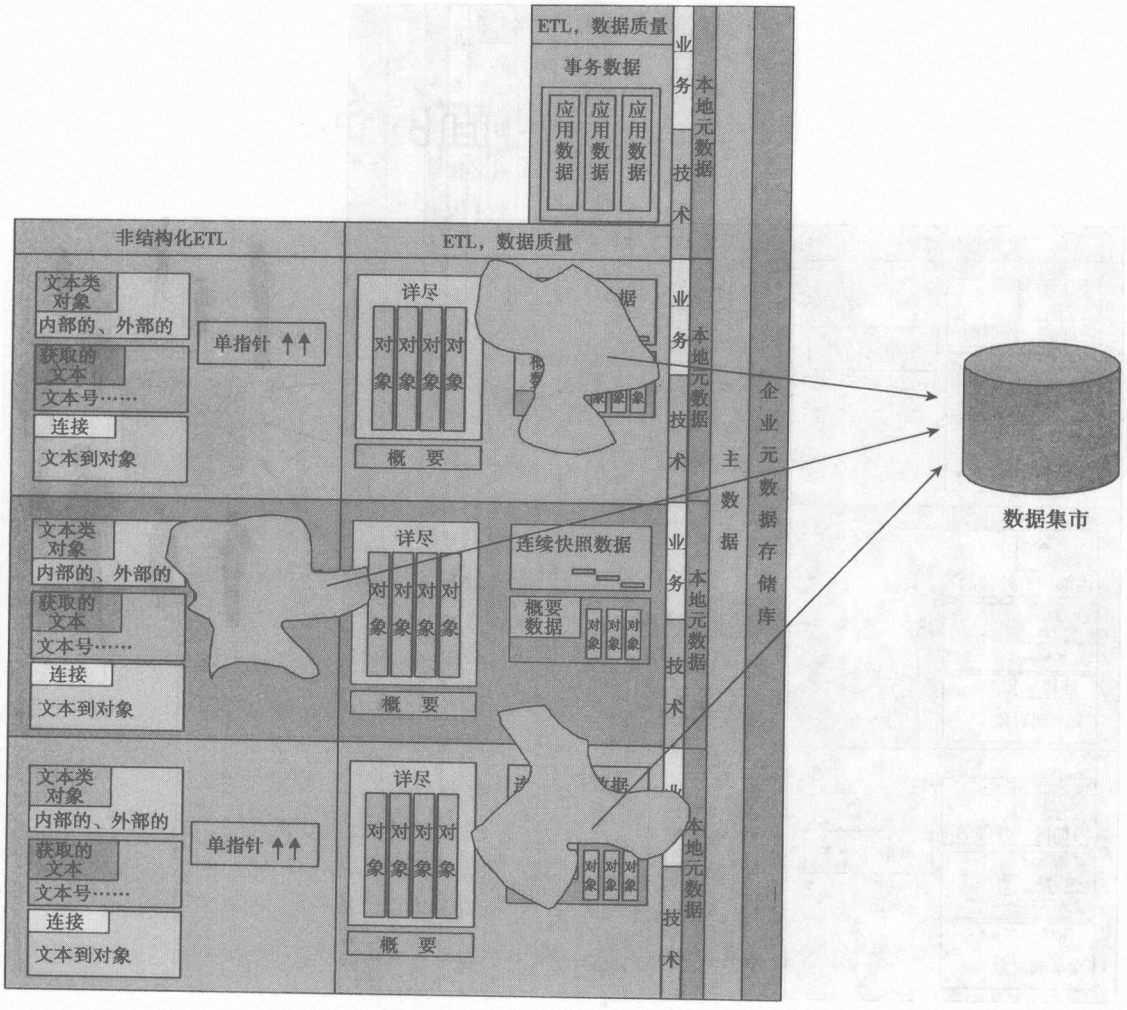


图 21-1 数据集市来自于 DW2.0 环境

自己的数据和处理持有所有权的方法。

数据集市盛行以及数据从主 DW2.0 环境中分离出来有着很多充足的理由。图 21-2 描述的是不同组别的人以相似的方式来查看数据。

21.3 转换数据集市数据

这种处理发生在数据从 DW2.0 环境移至数据集市环境时，包括数据汇总、数据聚集、数据选择及过滤、字段及其他数据属性的重组。

图 21-3 描述了在 DW2.0 环境中找到的详细数据换转成数据集市结构时发生的活动类型。

判断什么时候将分析过程由企业数据仓库移入数据集市中有意义是数据仓库构架师面临的最有趣的问题之一。答案是，当许多人以相同方式查看数据并且做大量查询时，创建数据集市就有意义了。

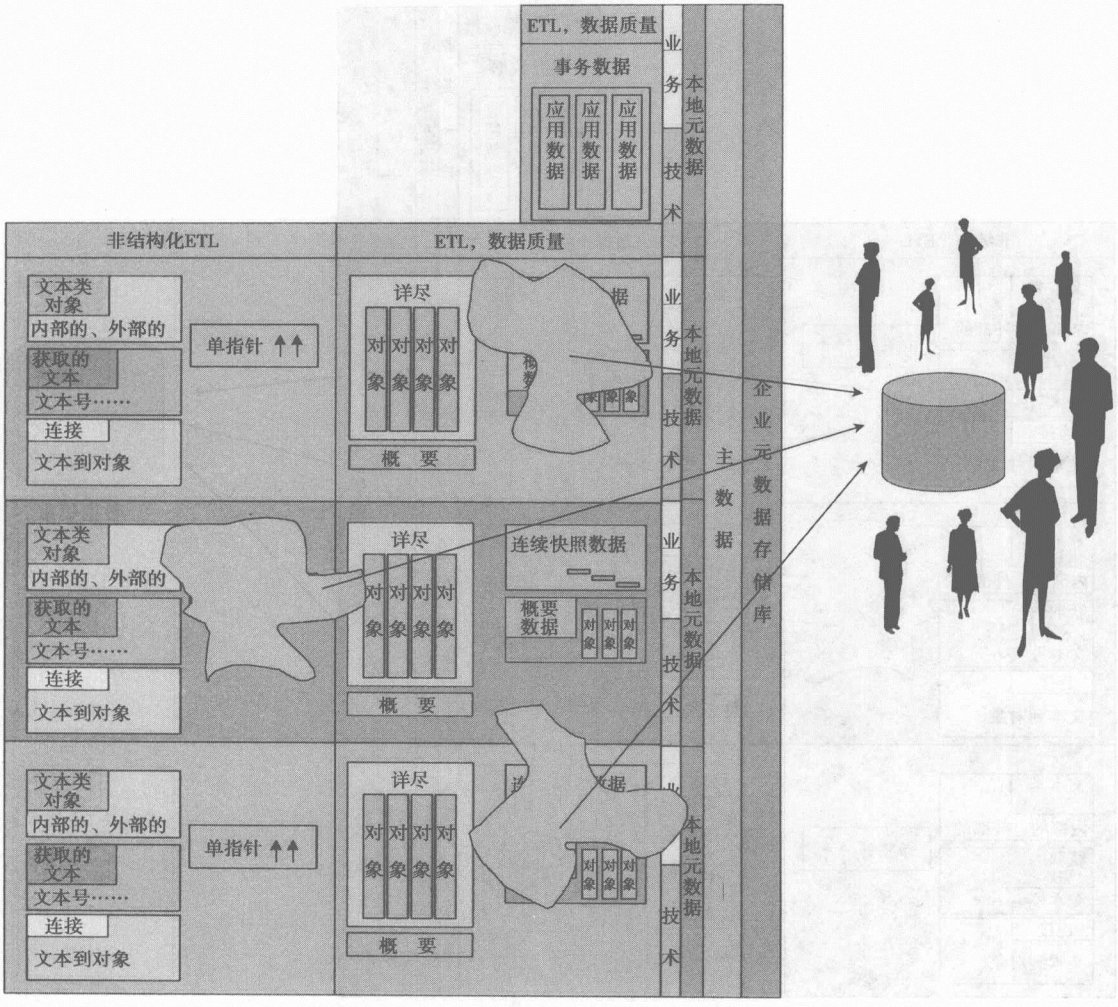


图 21-2 数据集市为那些以相近方式查看数据的人服务

21.4 监视 DW2.0

数据仓库监视器是获知何时应添加一个或多个数据集市的最佳方法之一。图 21-4 描述了一个数据仓库监视器检测数据仓库中正在进行的活动。当检测到连续的使用形式时可能就该建立数据集市了。

数据集市在建成后通常都是自治的，终端用户几乎能用他们的数据集市做任何想做的事。

21.5 在数据集市间移动数据

将数据直接从一个数据集市移动到另一个数据集市在构架上还不健全。图 21-5 指出，几乎所有情况下，都无法将数据直接从一个数据集市移动到另一个中。

如果有必要在两个或更多数据集市间共享数据，那么应将该数据放置在 DW2.0 企业数据仓库环境中。

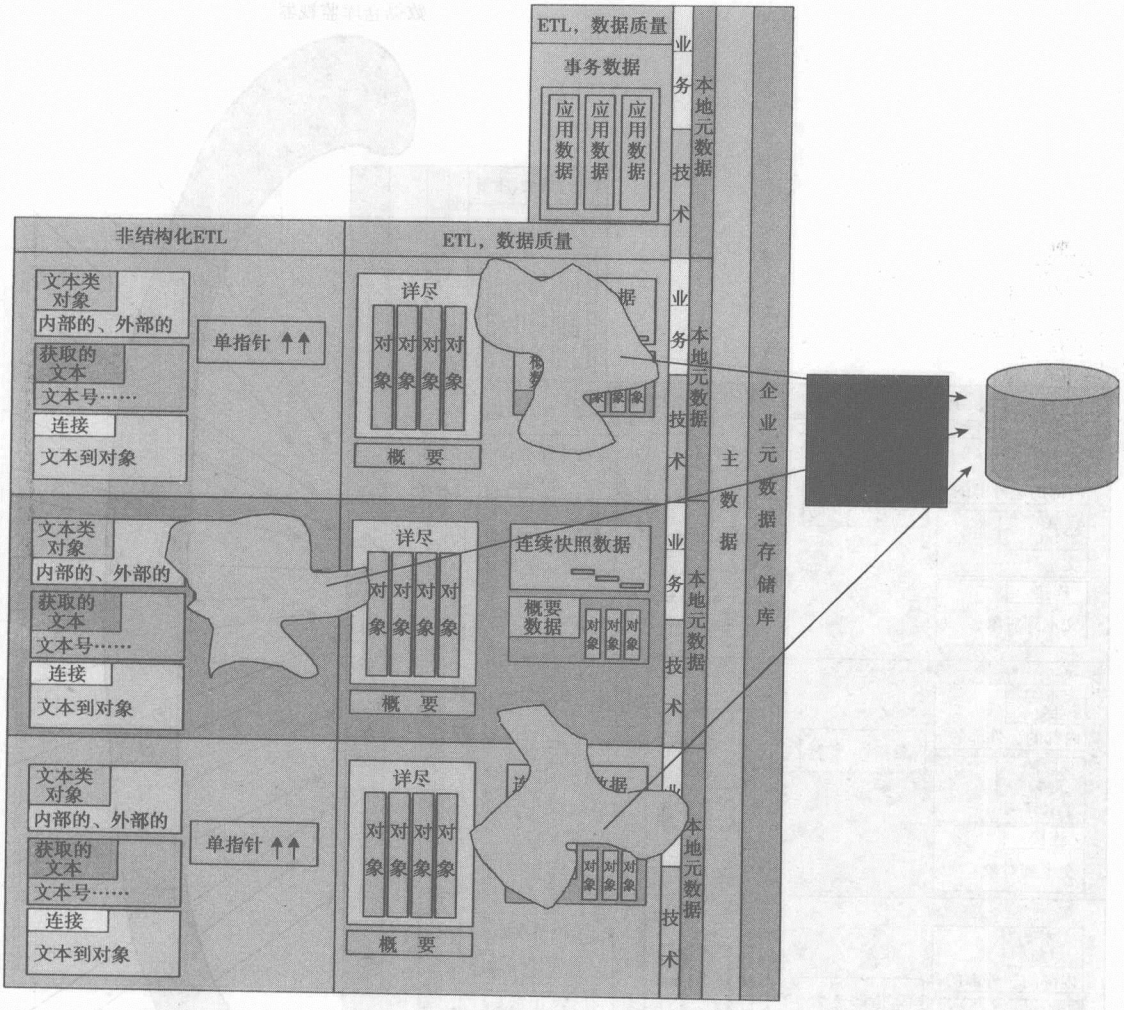


图 21-3 来自 DW2.0 的数据经过汇总、过滤、整合及重级，为建立数据集市做准备

21.6 不合格数据

期望所有数据都能完全输入到大型、复杂的企业数据仓库环境中是不合理的。图 21-6 描述了这一问题——对 DW2.0 环境中的不合格数据应该怎么处理？

应该做的第一件事是试图确认不合格数据的来源。如果能找到来源，下一步就是修正这一数据源。图 21-7 指出，第一步要找出不合格数据是如何进入到 DW2.0 的。

21.7 用以平衡的条目

修正一个将不合格数据传送到数据仓库的过程并不能解决如何处理已经进入数据仓库的不合格数据的问题。

修正 DW2.0 中不合格数据的一种方法是找到不合格数据并且构造一个“平衡”条目。如果发现系统中有一笔错误的数据条目 $23.61，那么再构造另一个等于 - $23.61

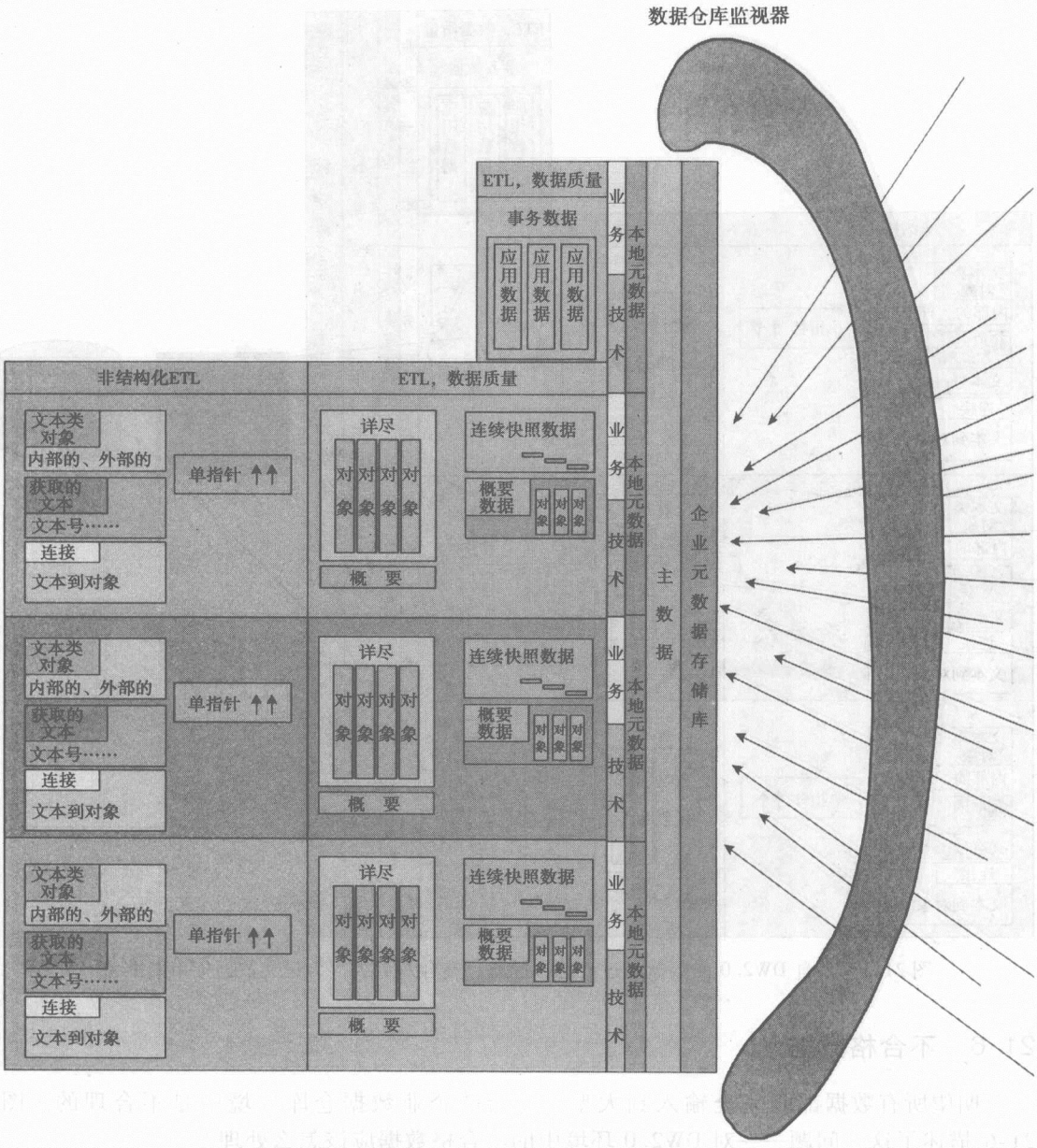


图 21-4 为了决策何时适宜建立数据集市，需监视对 DW2.0 环境的访问

的条目即可修正该数据。这种方法保持了账目平衡，并且留下检查跟踪。但是，这种方法仅限于调整有限数据并且能够确定错误数据的情况。

图 21-8 描述了平衡条目方法。

21.8 重新设置值

不幸的是，很多情况下，并不能找到数目确定的错误数据并为之建立平衡条目。这种情况下要强制建立一个条目来“重新设置”某个记录中的值。

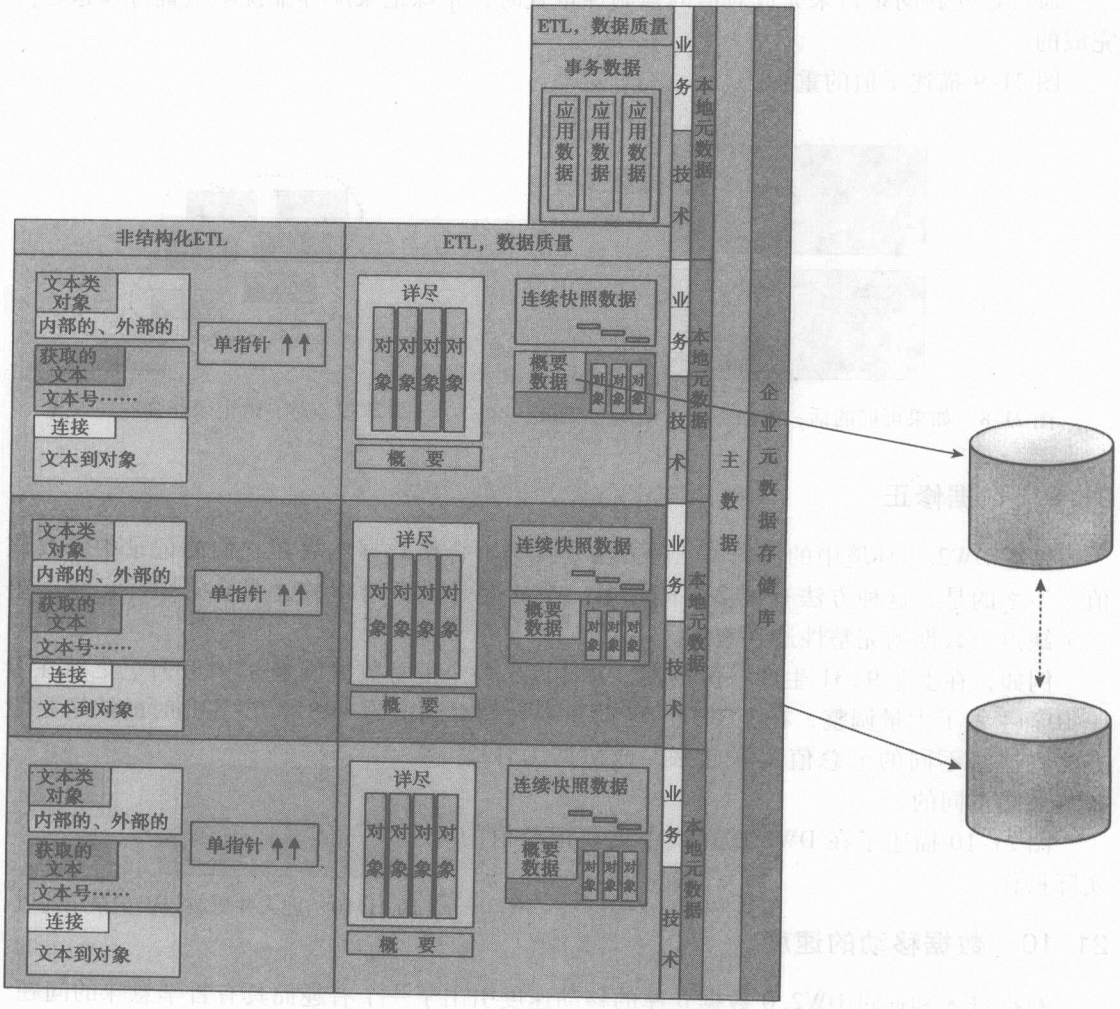


图 21-5 几乎所有情况下，在数据集市间分享数据的可操作性都非常差

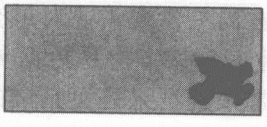


图 21-6 如何处理不合格数据

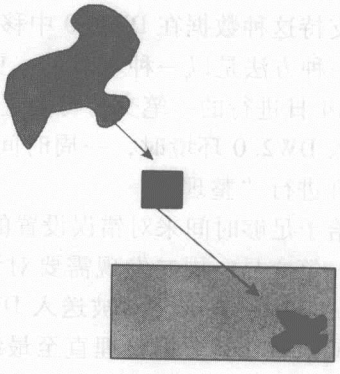


图 21-7 找出不合格数据如何进入 DW2.0 环境的原因并在 ETL 或其他过程中作修改

通过建立新的条目来完成对值的强制性重置时，跟踪记录应详细说明重置过程是如何完成的。

图21-9描述了值的重置。

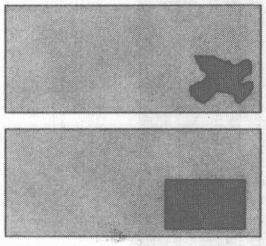


图21-8 如果可能的话，添加一条“平衡”记录

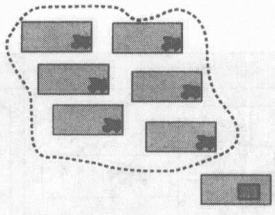


图21-9 对一组记录作强制性调整

21.9 数据修正

修正DW2.0环境中的值的第三种方法是找到不合格数据然后在它们的记录中修改该值。不幸的是，这种方法有很多缺陷。第一个问题是没有清晰的、明显的跟踪记录，第二个缺点是数据的完整性遭到破坏。

例如，在上午9:31生产一份报表，并计算出一个汇总数据值是$5918.91。之后在上午10:14做了大量调整，在上午11:57时重新计算得到值为$4817.73。此时的问题是没办法将这些不同的汇总值关联起来，或协调为什么这些值是不同的。

图21-10描述了在DW2.0环境中更换记录值的实际操作。

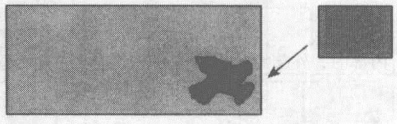


图21-10 进入并更改错误记录中的值

21.10 数据移动的速度

数据进入和通过DW2.0数据仓库的移动速度引出了一个有趣而具有哲学意味的问题。一些学派认为数据应尽可能快速地在整个DW2.0数据仓库中移动。换句话说，如果上午7:13产生一笔交易，那么在7:14数据就应进入并在交互区反映出来。图21-11指出，有许多人支持这种数据在DW2.0中移动时“尽可能快”的速度。

另一种方法是以一种较慢的、更慎重的方式在整个DW2.0环境中移动数据。例如，在1月14日进行的一笔交易可能直到1月21日数据才能进入DW2.0环境，这就意味着数据进入DW2.0环境时，一周时间已经过去了。这种数据在等待移入DW2.0环境的过程中允许进行“整理”。

当给予足够时间来对错误设置的数据进行调整时，数据会进行整理。例如，假定周一进行了一笔交易，周二发现需要对该交易做出调整，而在周三发现需要对该交易进行再次调整，最后周五该交易被送入DW2.0环境，那么这种不急于将交易数据传入DW2.0的方式就给将交易数据整理直至最终状态提供了可能，带来的结果是更精确的数据以及对于DW2.0而言更简单的处理。

图21-12展示了在传入DW2.0环境之前允许数据进行整理的过程。

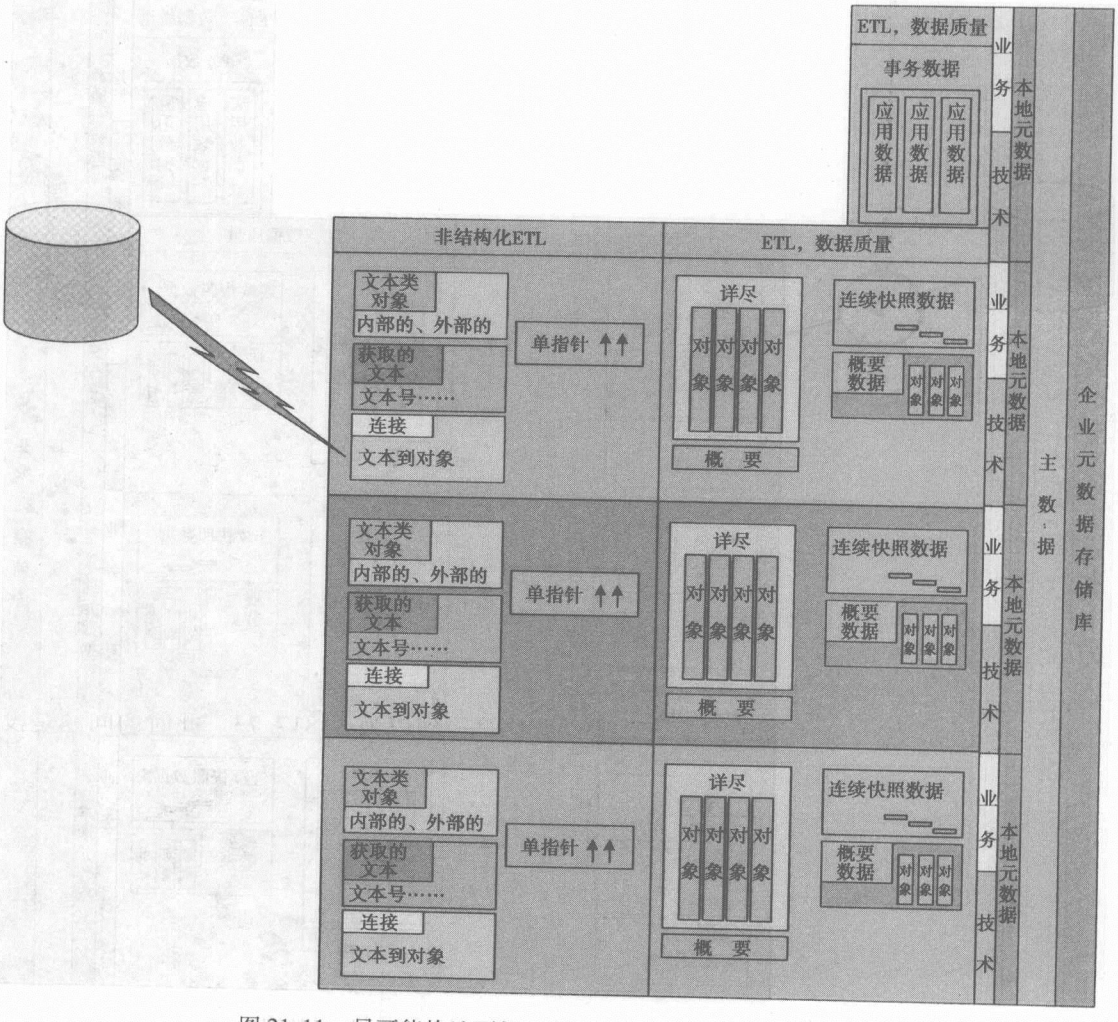


图 21-11 尽可能快地更新 DW2.0 中的数据——一种方法

21.11 数据仓库工具

数据仓库工具（DWU）是采用一些或所有数据仓库处理，并且透明地替换现有的一些或所有数据仓库基础设施的设备。对于运用数据仓库工具有许多充足的理由，包括性能、成本以及延长 DBMS 的许可期限等。“Dataupia” 是数据仓库工具的一个不错的例子。

以下实例说明了为什么数据仓库工具大有好处。如图 21-13 所示，看看“标准的”数据仓库处理环境。

本图描述了一个终端用户与 SAP 之类的技术环节直接交互，而 SAP 又与例如 Oracle 之类的 DBMS 直接交互，Oracle 与传统的例如 EMC、IBM 或 Hitachi 之类的 SAN 技术直接交互。

如图 21-13 所示，随着时间的增长，传统环境中的大量数据量开始增长并且可能变得非常巨大。图 21-14 描述了随时间变化传统数据仓库会发生什么。

在这种环境中数据如此固定地增长的基本原因有三点：

- 数据均以低粒度级收集得到。

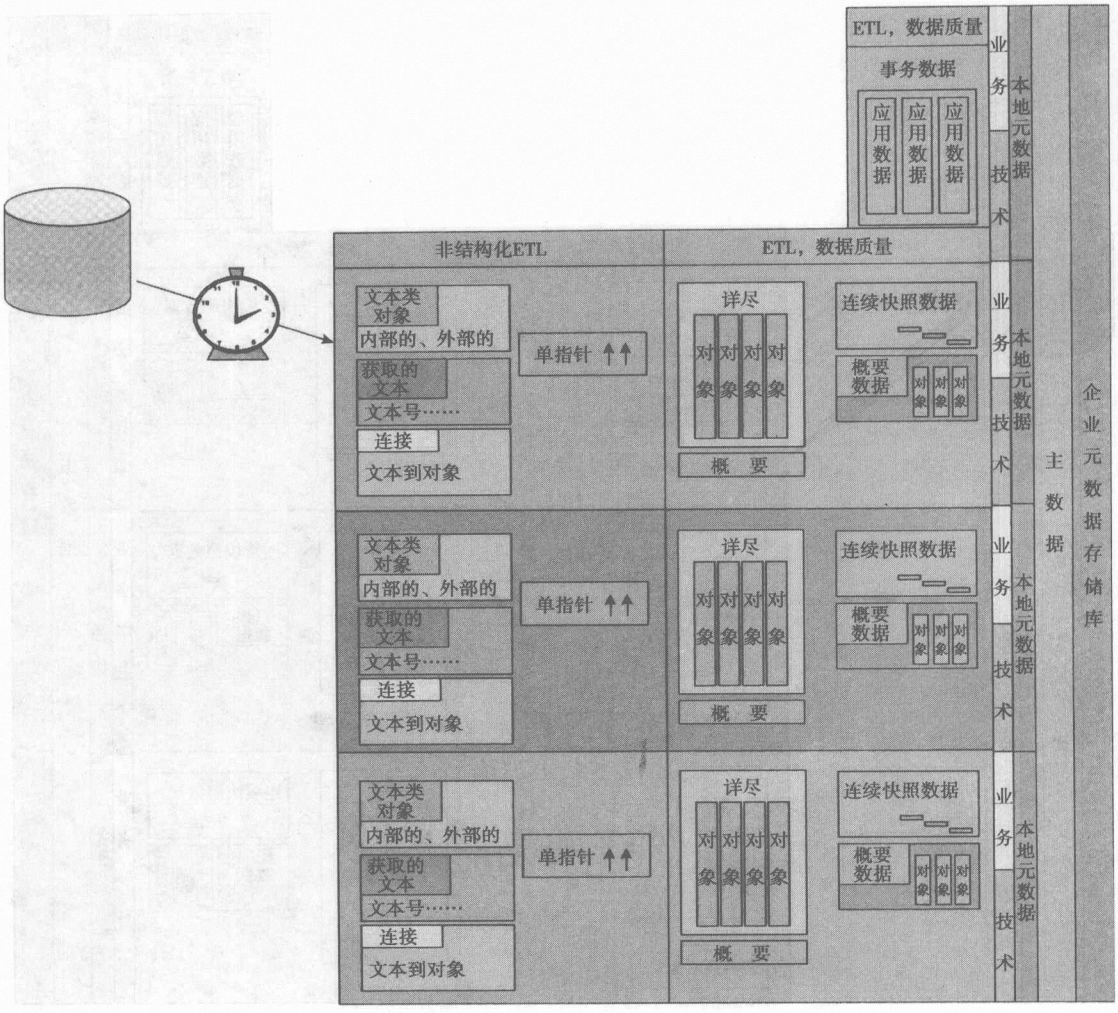


图 21-12 在移入 DW2.0 之前允许数据时间进行“整理”

- 数据是过去收集的。
- 数据是从多种多样的数据源收集并整合的。

数据增长有很多后果，一个最大的后果是数据以及支持它的基础设施的成本会大幅增长。图 21-15 指出，作为数据量管理功能，基础设施的成本会升高。

成本不仅仅是增长——而且是大幅增长。

在数据仓库处理中，存储成本是个有趣的因素。在建造和发展数据仓库的头两三年，存储成本几乎不重要。但是当数据仓库成熟后，数据仓库其他方面的成本减少了，而存储成本却增长了。而且，不仅存储成本增长，用于存储的基础设施成本也相应增长，有处理成本、软件许可成本以及销售渠道成本。另外，在获得并实现存储后会不断地产生操作成本。当针对这些因素而考虑实际的存储成本时，它仅仅是总的存储成本中的一小部分。

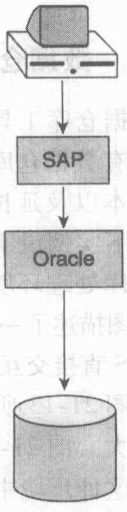


图 21-13 传统 DW2.0 环境

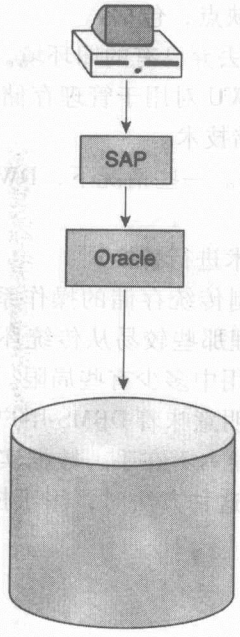


图 21-14 DW2.0 变旧后会发生什么

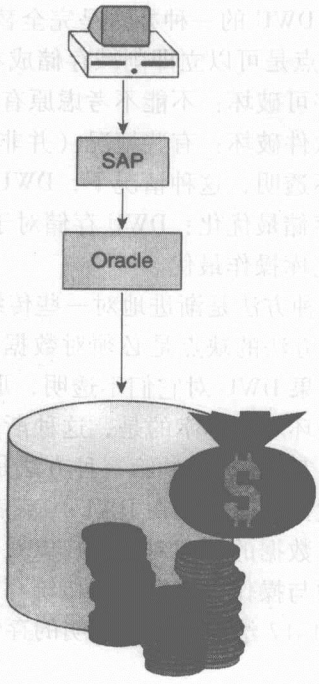


图 21-15 基础设施的成本

不幸的是，存储及存储基础设施的成本是不可避免的。一旦某机构受困于某一处理，那么它就必须继续下去，并且很长时间。

但是，组织机构需要管理预算，每年支出的主要增长不能总是不明确，而组织机构想要找到管理预算的方法也是很自然的。因此，他们求助于数据仓库工具来帮助他们管理预算以及他们的数据仓库环境就很正常了。

数据仓库工具是一种为传统的 SAN 存储的一小部分成本存储和管理数据的方法。

图 21-16 描述了一个传统存储与 DWU 相结合方式管理的数据仓库。

本图表明一部分数据受传统存储方式管理，另一部分则受 DWU 管理。这样分开管理的效果显著地表现在大大降低了数据仓库日常的基础操作成本。

从细节看，增加 DWU 后操作数据仓库的成本可能会极大减少。例如，假定某企业有一个数据量为 10TB 的数据仓库，每年的操作预算是 $10 000 000，再假定该企业通过加入 DWU 将其数据仓库存储需求减半，那么它每年的操作成本如下：

5TB 数据使用传统存储技术——$5 000 000

5TB 数据使用 DWU 存储技术——$500 000

总的操作成本——$5 500 000

总共节省——$4 500 000

将数据移入 DWU 大大削减了传统存储技术连续不断的操作成本。执行 DWU 并不像把电源插头插入电源插座那么简单，DWU 技术的配置有多种策略，每种配置都各有其优劣。

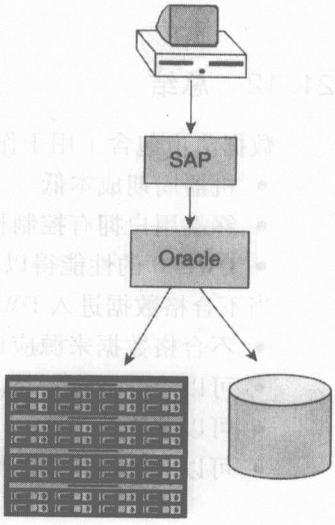


图 21-16 数据分割

运用DWU的一种办法是完全替代传统技术——DWU转入而传统技术转出。这种替代策略的优点是可以立即削减存储成本，同样，它也有一些缺点，包括：

- 许可破坏：不能不考虑原有的合同及许可，而轻易丢弃已有的旧环境。
- 软件破坏：有些情况（并非Dataupia）下，一些DWU对用于管理存储操作的控制不透明，这种情况下，DWU无法彻底取代传统存储技术。
- 存储最优化：DWU存储对于OLTP操作不是最优的。一些情况下，DWU只对数据仓库操作最优。

另一种方法是渐进地对一些传统数据仓库中的存储技术进行替换。

这种方法的缺点是必须对数据透明，DWU必须与控制传统存储的操作系统及DBMS兼容。如果DWU对它们不透明，那么它必须访问并且管理那些较易从传统环境中分离出来的片段环境。不幸的是，这种渐进地、分离的方法在应用中多少有些局限。

运用DWU技术的第三种方法是透明地配置DWU。透明意味着DBMS用户不知道数据位置，数据可能存储在DWU中或者是传统存储中，系统根本不在乎。数据实际的物理位置以及对数据的管理对于用户以及DBMS是透明的。采用这种方法时，用于控制DWU的软件必须与操作系统及管理传统存储的DBMS兼容。

图21-17给出了这种透明的存储方式。

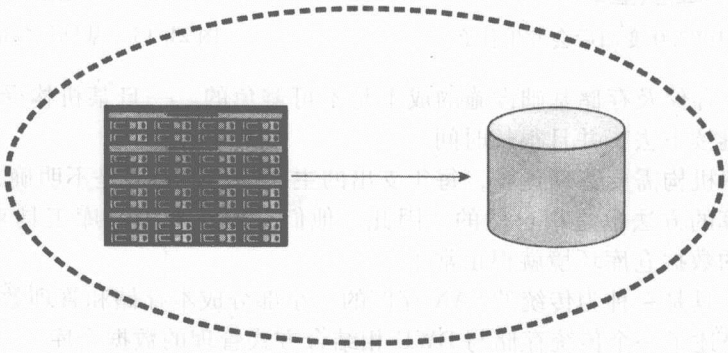


图21-17 数据库透明度

21.12 总结

数据集市包含了用于作决策的部门数据。有若干理由支持建立数据集市：

- 机器周期成本低。
- 终端用户拥有控制权。
- DW2.0的性能得以提升。

当不合格数据进入DW2.0环境中时，

- 不合格数据来源应该明确并且得以修正。
- 可以建立平衡条目。
- 可以重置值。
- 可以对数据进行实际修改。

第 22 章　DW2.0 环境中的处理

DW2.0 环境的显著特点是在不同的区域能找到各种类型的数据。从许多方面说，是数据与区域定义了 DW2.0。但从系统构架师的角度而言，DW2.0 不仅仅是一个数据构架。另一种方式是通过理解在各种环境或区域下找到的过程来理解 DW2.0。

DW2.0 环境中存在多种类型的事务和过程，也许这些事务中最简单的就是一个简单的数据请求了。图 22-1 给出了一个简单的数据请求。

一个简单的数据请求是想要找出一两行数据，然后以交互的形式显示它们。这种简单的事务占用很少量的系统资源并且在逻辑上非常简单。这种事务经常存在于在线环境中，因为当系统正在执行这类事务时很容易得到优良的性能。

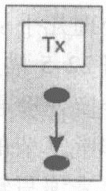


图 22-1　简单的访问事务

这里所描述的事务是预先定义的。因为其逻辑是预先确定的，所以终端用户只不过是运转使其执行事务。

复杂事务是简单事务的一种变形。复杂事务通常比简单事务查看更多行的数据，且包含了相当多的逻辑，以及一些简单事务中所不包含的东西。如果复杂事务的执行不需要太多数据，那么它就可以自由地混入交互区的工作流中，而不会严重降低性能。

图 22-2 显示了一个复杂事务。

复杂事务几乎总是预先确定的，它们仅仅是通过终端用户来运转以执行。

另一种简单事务的变形是一种基于特定基础上的事务。图 22-3 描述了这种特定事务。

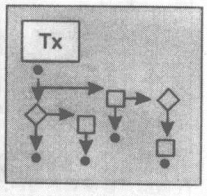


图 22-2　复杂事务

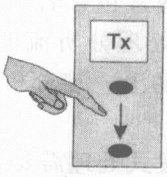


图 22-3　特定事务

特定事务通常非常简单，不会存在复杂的逻辑。特定事务通常也不查看太多数据，但偶尔终端用户也会递交一个需要查看大量数据的特定查询请求。当运转一个要查看大量数据的特定事务时，性能就会受到影响。

出于这样的原因，在交互环境中往往没有太多的特定事务，而在整合环境中特定查询请求才更普遍。

在很多情况下，会在数据集市环境中发现特定查询请求，而这些请求往往是由商业智能软件产生的。实际上，除了参数外，终端用户不向商业智能软件输入任何东西。一旦写入参数后，就由商业智能软件生成查询请求。

另一种查询类型是访问查询。访问查询与简单的访问查询的不同在于访问查询往往要访问大量的数据。

图22-4描述了访问查询。

访问查询的逻辑往往非常简单，然而它所涉及的数据量可能会相当大。

分析人员使用访问查询来扫描全部数据。有时候会出现的仅查看一两行数据的情况是不能提供所需要的信息的。由于要访问大量数据，访问事务通常不会在交互区中执行。如果要在交互区执行，那也只能在对系统的整体性能没有损害的空白时间段内执行。相反，访问事务更常见于在整合区和存档区中执行。另外，访问查询很少在近线环境中执行。

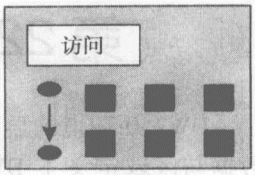


图22-4 访问查询

DW2.0环境中的另一种常见的处理是转换处理。转换处理对整体数据进行访问、改变以及写入新文件。在处理高峰期时，转换处理几乎从不在交互环境中运行。

图22-5描述了一个转换处理。

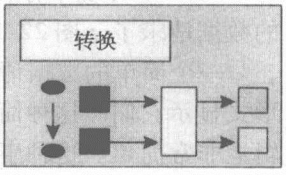


图22-5 转换处理

转换处理通常有着复杂的算法。有些情况下，转换处理还包含非常复杂的过程。由于这个原因，一般都是在预先定义的基础上来编写转换处理，而其他形式都不常见。换种说法就是，特定的转换处理在本质上绝不是特定的。

元数据是转换处理的一个副产品。转换处理所执行的转换也都是由元数据形成的。因此，作为处理的文档，元数据被写出并对DW2.0环境下的许多人都非常有用。

转换处理既适用于结构化数据也适用于非结构化数据。

还有一种处理是统计处理。统计处理对于大量数据的数学分析非常有用。由于几乎所有情况下的统计处理都需要访问大量数据，因此当在线响应时间是一个要考虑的因素时，就不能运行统计处理。

图22-6展示了一个统计处理。

统计处理通常包含着复杂的处理逻辑。它们往往是所谓的启发式处理分析流的一部分。在启发式处理中，只有在紧接着的上一步分析完成后，下一步分析工作才比较明显。

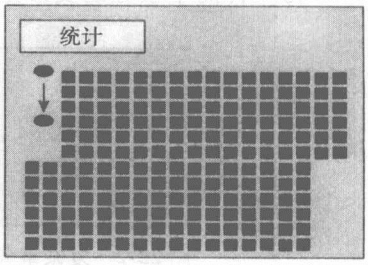


图22-6 统计处理

因此，启发式处理需要一种特定的处理过程。

DW2.0环境的不同部分通常执行不同类型的处理。图22-7给出了交互区中的处理类型。

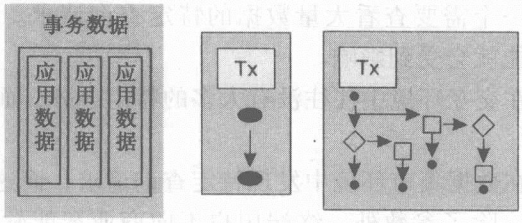


图22-7 交互环境中的工作流大部分看起来像什么

交互区中有简单事务和复杂事务，没有统计处理，没有访问处理，只有能在不存在资源冲突的地方以有条不紊的方式运行的事务。换言之，交互区环境的工作流用于少量的、

快速运行的以及有良好秩序的事务。除去这些情况，其他情况都会中断事务流并且对性能产生消极影响。

整合环境能执行各种处理。图 22-8 给出了整合环境中能够执行的处理类型。

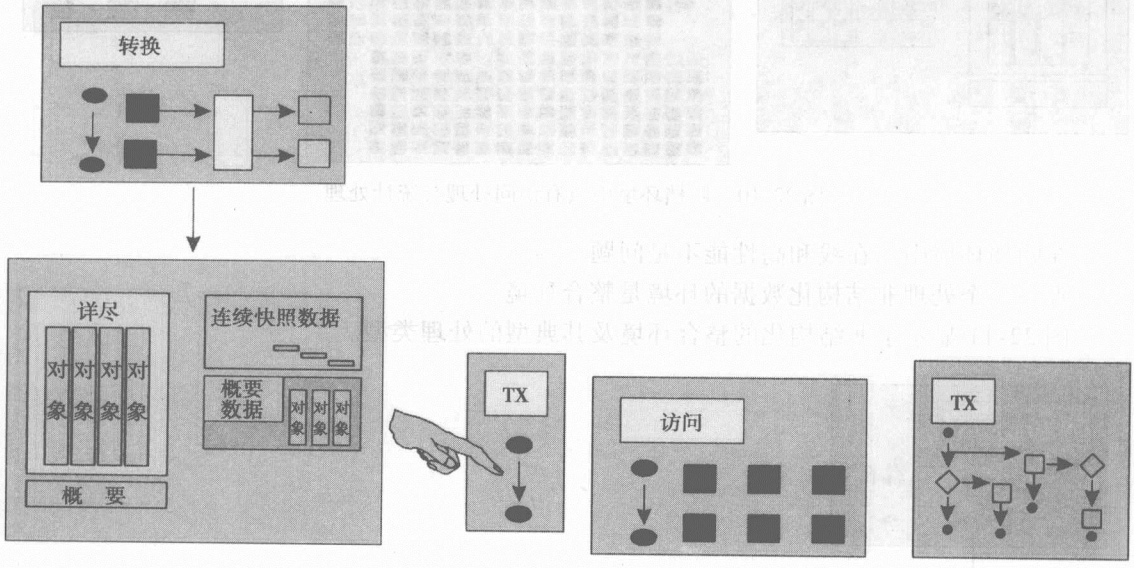


图 22-8 整合结构化环境中的处理

从图 22-8 中可以看到，当数据进入整合环境时进行转换处理。环境一经创建，就执行特定处理、访问处理以及复杂处理。

整合环境中执行处理的最终结果是混合的工作量，正因如此，系统的整体性能不够稳定。

图 22-9 给出了近线处理过程。

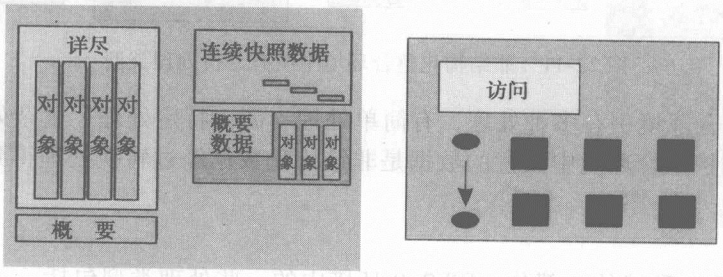


图 22-9 近线环境中很少或没有终端用户处理

实际上在近线环境中很少执行处理，大概只有两种处理即访问处理和替换处理。替换处理是一种专用处理，处理那些从近线环境中获取的并且放置在整合区的少量数据。

归档环境中实际上很少执行处理，然而在归档环境中执行的操作往往是资源非常密集的处理。图 22-10 给出了归档环境中的处理类型。

归档环境中的普遍处理是统计处理和访问处理。如果已建立被动索引，那么归档环境中的处理通常会很有效。但如果未建立被动索引，那么不得不在归档环境中对全部数据进行扫描。

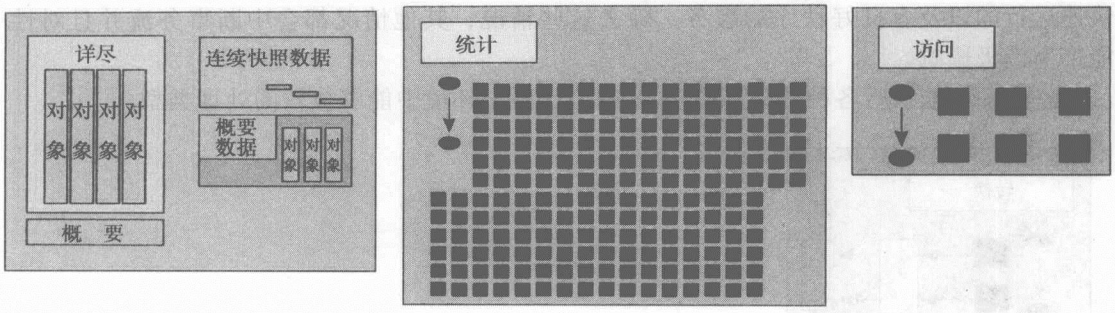


图 22-10 归档环境中只有访问处理与统计处理

在归档环境中，在线和高性能不是问题。

唯一一个处理非结构化数据的环境是整合环境。

图 22-11 显示了非结构化的整合环境及其典型的处理类型。

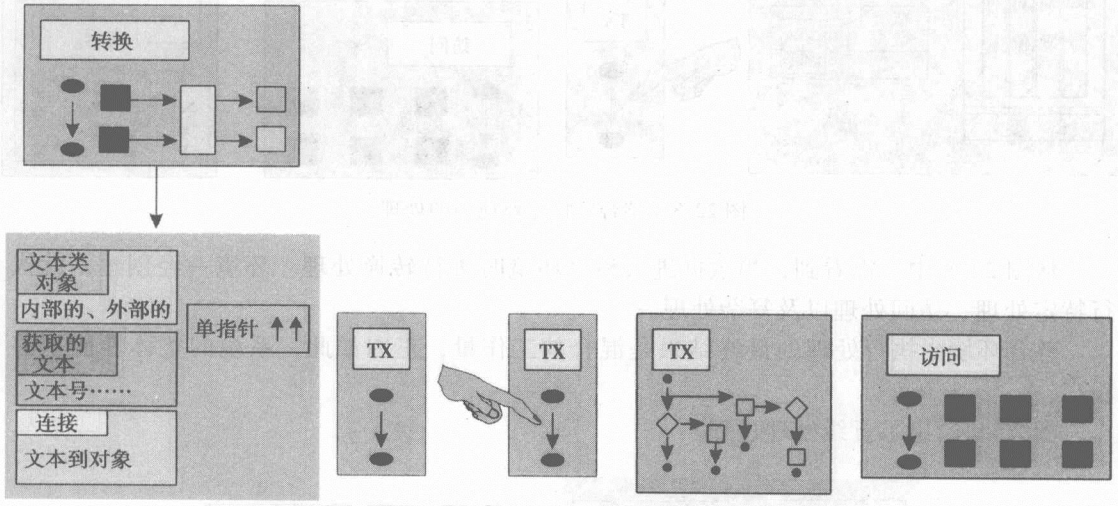


图 22-11 非结构化整合环境中所有分析处理类型

非结构化整合环境中有多种处理，有简单处理、简单特定处理、复杂处理以及访问处理。另外，非结构整合环境中放置的数据是非结构化数据经过转换处理得到的。

总结

处理是 DW2.0 环境的一部分。DW2.0 环境中的一些处理类型包括：

- 简单事务。
- 复杂事务。
- 转换处理。
- 访问处理。
- 统计处理。

由于这些处理中的数据以及各区的性能特征，不同类型的处理在 DW2.0 环境中的不同位置有着密切的关联。

第23章　管理DW2.0环境

DW2.0环境是一个非常复杂的环境，需要很长的时间来构建。DW2.0环境涉及企业的很多部分：日常操作、管理、战术战略决策甚至是董事会。DW2.0环境也包含很多方面，如技术、商务、法律、工程以及人力资源等。因此，DW2.0环境是一个长期管理问题，需要小心去经营管理。

这一章涉及的是在DW2.0环境下一些经营管理方面的问题。

23.1　数据模型

DW2.0环境中的知识核心是数据模型。数据模型用来描绘如何用技术来满足业务需求。在很长一段时间里，数据模型都是用来指导不同开发者的开发工作的。如果能合理地运用数据模型，一部分开发接着另一部分开发的过程就像拼接一幅巨型拼图。换种说法，如果没有数据模型，在DW2.0环境下要想协调多个长期的多人开发项目是一项几乎不可能完成的任务。

数据模型包含了多种不同的层次，有高层、中层、低层。第一步（也是最难的一步）是定义数据模型的整合范围。整合范围之所以很难被定义是因为它绝不是静止的，而是持续变化的，并且每一次改变都影响着数据模型。

当这个范围变化得太频繁和太快时，企业将遭受“范围蠕变”。

高层数据模型很少需要随着时间的推移而维护，而中层数据模型和低层数据模型则会受到企业中的长期变化的明显影响。随着时间变化，中层数据模型中的主键、数据关系、数据域、数据定义、属性，甚至是组属性都会发生改变。而每次改变的发生，相关的物理关系数据库也会随之改变。

数据模型管理的部分工作是为了确保数据模型每次改变都有相应的对数据仓库的新开发和新修改。其中，要确保的最大问题是：

- 不能引入数据模型中没有的新的基本属性，或者当要引用新的基本数据元素时，它们可以加入数据模型。
- 新开发者能够将数据模型看作是前进的促进剂，而不是前进的壁垒。
- 对DW2.0做的新修改要遵从数据模型。

需要特别注意的是，数据的属性组和主键/外键对数据间的一致性非常重要，而数据模型的其他方面就没有那么重要了。

另外，主数据派生出来的数据不需要遵从数据模型。

23.2　构架管理

除了为遵循数据模型而需要的一个趋向数据模型的管理机构外，还必须有一个一般的构架机构来管理DW2.0架构。构架管理趋向于对构架进行长期的监控。接下来给出几个构架管理需要注意的地方。

23.2.1 确定什么时候需要归档区

大多数环境都不需要立即创建归档环境，而常常是过一段时间后才创建它。构架管理为何时及如何创建归档环境提供了指导。构架管理决定了归档环境的许多方面，例如：

- 数据进入归档环境的时间。
- 数据在归档环境中停留多长时间。
- 将数据移出归档环境的准则。
- 归档平台。
- 归档环境的数据库设计。
- 被动索引是否将被创建。
- 是否创建被动索引。
- 归档数据的粒度级别。
- 其他内容。

23.2.2 确定是否需要近线区

如果需要近线区，那么构架管理就会确定一些重要的参数。例如，何时将数据移入近线区、整合区和归档区；要存储哪些元数据；近线区使用什么平台，等等。随着时间推移，对近线区的需求也会发生变化。在最初设计时，可能很明显就可以看出根本不需要近线区。但过一段时间后，决定需求的因素可能发生变化。因此，我们总有一天可能要用到近线区，而决定是否需要近线区只是构架管理的工作。构架管理员能够做的决定包括以下几种：

- 是否需要近线存储器。
- 数据进入近线存储器中的标准。
- 近线存储器所需的平台。
- 即将存储的元数据。
- 数据移出近线存储器的标准。

交互区是DW2.0环境中另一个构架管理员所关注的区。有些企业中有交互环境，而有些却没有。构架管理员主要解决如下的问题：

- 是否需要交互环境？
- 如果存在一个交互环境，那么它的响应时间是否合适，是否合乎所有服务标准协议（SLA）的要求；可用性是否适当，是否合乎所有SLA要求；交互环境是否可用于任何需要完成的报表；是否满足容量要求？
- 当数据移出交互环境时，其是否被适当地整合？
- 假如要将遗留数据读入交互区，那么是否已将其适当地整合到应用当中？
- 交互区工作在什么平台？

构架管理员的另一项任务是确保不存在从一个数据集市到另一个数据集市的数据流。当管理员发现这种情况时，他/她应当重定向一个数据集市的数据流，使其流向DW2.0环境，然后再从DW2.0返回到另一个接受数据的数据集市。

构架管理员还有一项任务就是确保能够进行适当的监视并对监视结果进行适当的解

释。DW2.0环境中需要很多监视活动。例如，需要监视交互区中的交易和响应时间，还需要监视DW2.0环境其他部分的数据及其使用情况。

对DW2.0环境下的监视，需要考虑以下几个问题：

- 交互区中的交易是否正在被监视？
- 交互区的可用性是否正在被监视？
- 整合区中的数据使用是否正在被监视？
- 休眠数据确定了吗？
- 监视器会浪费大量系统资源吗？
- 何时对监视结果进行检查？

监视整合区数据使用的最重要的结果是决定什么时候创建一个新的数据集市。管理员查找整合区中重复的数据使用模式，当相同结构的数据请求出现的次数足够多时，就表明需要数据集市。

以上是DW2.0环境中的一些构架管理活动。但是DW2.0环境其他一些方面也同样需要构架管理。

毫无疑问，构架管理员需要掌握的一项能力是理解构架。如果让一个不知道构架的含义且不知道构架都该考虑哪些的人来当构架管理员，那肯定是白费工夫。

构架管理的另一个重要部分是管理DW2.0中的ETL处理。DW2.0中的第一种ETL进程是传统的对从应用源中得到的数据的整合。此时需要监视的问题包括：经过ETL处理的数据流动，数据转换的准确度，这些转换对分析机构的可用性，以及转换的速度、容易度等。另外一类ETL工具是文本转换，通过文本转换可以将非结构化数据转入DW2.0中的数据仓库中。此时的管理问题包括：进入DW2.0的数据量，使用的整合算法，DW2.0中的数据类型等内容。要注意的是，两种类型的ETL转换是完全不同的。

23.3 元数据管理

元数据是DW2.0环境中最重要的一个方面。由于种种原因，元数据的管理是一项单独的任务。其中一些原因如下：

- 元数据的捕获和管理工具的发展大大滞后于其他技术。
- 之前的元数据管理并不成功，失败次数多于成功次数。
- 相比DW2.0环境其他方面的业务案例，有关元数据的业务案例需要更多的关注。

当然，还可能存在更多的原因，使得元数据管理成为一个敏感问题。

问题是需要使用元数据来将DW2.0环境的不同部分有意义地结合在一起。也就是说，如果没有一个有内聚性的元数据基础结构，DW2.0的很多不同部分将无法协调它们之间的工作。

元数据管理需要包含很多方面，其中包括：

- 元数据的原始捕获。
- 元数据的编辑。
- 在DW2.0环境中的适当时间和地点使元数据可用。
- 元数据的持续维护。
- DW2.0环境中不同地方的元数据分布。

- 元数据的进一步扩展。
- 元数据的归档。

除了以上这些考虑，元数据管理员还要确定以下重要内容：

- 元数据的运行平台。
- 捕获和存储元数据所采用的技术。
- 展示元数据或使元数据可用所采用的技术。

元数据的一个问题是它的短暂性。跟结构化数据不同，元数据存在于多种形式和结构中，因此很明显它不像其他数据形式一样具有稳定性和适应性。

元数据还有一个主要问题是它有多种数据形式。其中元数据的两种基本类型是：

- 业务元数据
- 技术元数据

通常，技术元数据比业务元数据更容易识别和捕获，这其中的原因大家早就知道了。实际上业务元数据早就被看作是信息领域的一部分，但从厂商、产品、技术等方面都没有正式地定义业务元数据。所以相比较业务元数据，技术元数据更容易被找到并确定。

23.4 数据库管理

DW2.0 中另一个至关重要的方面是数据库管理，它要完成数据库的日常关注和管理。这是一项技术工作，需要了解如何存储数据库，如何恢复丢失的事务，如何判断何时丢失事务，当数据库关闭时如何备份数据库等问题。

简而言之，当数据库出现问题时，数据库管理员负责完成对数据库的备份并使其运转。

数据库管理的挑战之一是 DW2.0 环境所需的数据库管理活动的绝对数量。数据库及表是如此之多，以致于数据库管理员在任何一个数据库上投入大量的时间都是不可能的。因为它们的数量太多，而且每个数据库都非常重要，因此，管理员需要用工具来查看这些组成 DW2.0 环境的众多数据库和表的多个方面。

DW2.0 中对数据库进行管理需要考虑以下问题：

- 为 DW2.0 环境中数据库管理的监视而选择工具。
- 为 DW2.0 环境中的数据曲线及对其所带来痛苦的预防而选择工具。
- 确保在需要时能使用这些工具。

通常，数据库管理是个一周 7 天、每天 24 小时的工作。负责数据库管理的人应该是在所有时间都随叫随到，并当出现问题时能够给出怎么去做的建议。尤其是在交互环境下，当数据库发生问题时，数据库管理员要尽可能地主动，因为故障和停机都会让人对环境不满意。但主动处理是很困难的，因为数据库管理员所要应对的绝大多数任务都是有反作用的。

23.5 数据管理

近些年，管理和遵循准则已经成为一个大的问题，因此数据管理的角色也就成为一个重要话题。过去数据管理的工作仅仅是系统的数据输入和输出，而现在，数据的质量和准确性已变得非常重要。

在此构架中，数据管理已被提升到被公认为需要承担责任的位置上。

数据管理工作需要承担以下工作：

- 确定哪些数据元素构成了记录系统。
- 对这些数据元素的数据质量标准的规范说明。
- 这些数据元素的相关算法和公式的规范说明。

为了区分数据库管理员的职责和数据管理员的职能，需要考虑以下问题。当数据库出现故障并对系统不可用时，或是当性能下降并且出现一个整体的系统停机时，需要数据库管理员来处理；而当终端用户发现记录中存在错误值时，或当需要设计新的数据库以及考虑数据来源和数据转换时，就需要数据管理员了。

因此，数据库管理员和数据管理员负责不同的事情。通常，数据库管理员是技术人员，而数据管理员是业务人员。试图将数据管理员的工作看成是技术性的工作通常是不对的。

数据管理员的一些工作包括：

- 可以承担数据库的设计工作，尤其是设计中包含转换和映射的设计。
- 能够回答指定数据元素内容的相关问题。
- 讲解给业务分析人员都有哪些数据以及如何最好地解释这些数据。
- 确保能够准确地设计映射和转换。
- 描述如何完成算法和程序逻辑以能反映数据真正的业务含义。

一个大型企业通常有多个数据管理员，且多是由业务人员来承担这个角色。任何时候，每个原始数都有且只有一个与之对应的数据管理员。如果一个数据元素在任何时候没有或有多个数据管理员，都将会出现问题。

23.6 系统和技术管理

系统和技术管理是DW2.0环境的一个整体部分。DW2.0环境最终运行于多个平台之上。由于数据、处理以及对DW2.0不同部分的要求等都是多样的，所以只用一个平台来服务于整个DW2.0环境的情况是不常见的。相反，需要结合使用多种不同技术和平台以满足DW2.0处理的需要。

DW2.0中有的地方要求有很高的性能；有的地方关注于数据整合；有的地方要求能对数据进行长时间的存储；还有的地方则需要满足终端用户的分析需求。简而言之，就是存在着许多不同的标准来确定在不同地方DW2.0环境是否是成功的。

由于存在着许多不同的需求，所以没有一种单一的技术或平台能同时满足所有的需求也就不足为怪了。

因此，DW2.0中技术和系统的管理员需要充当多种角色，属于技术管理员的任务有：

- 保证技术兼容性，例如，确保数据能在不同环境下传送，系统的性能不受其他系统影响，数据能在所有的系统间整合，以及确保在整个环境中的可用性等。
- 确保对DW2.0中所有组件有一个长期的发展计划。

保证元数据在DW2.0环境的各组件间有意义地交换。

- 确保终端用户清楚地了解对于不同的处理，应用DW2.0中的哪些组件是合适的。
- 网络管理——确保整个DW2.0环境中能够且高效地进行通信。
- 定时——确保数据间能够以一种流畅无阻的方法相连接。
- 性能——确保整个DW2.0环境的性能是可接受的。
- 可用性——确保DW2.0中的各个组件在需要时能正常运行。

- 确保在终端用户需要时所需的元数据流是可用的。

技术管理员工作的一项重要内容是容量规划。技术管理员的工作在很多方面都类似于数据库管理员的工作。技术操作在很多时候是以交互模式进行的，且没有人喜欢一直被昨天已经完成的事情所烦恼。而这也正是技术员及数据库管理员都可能遇到的情况。

技术员想要摆脱交互模式工作的一个重要办法是进行适当的容量规划。并不是所有差错和问题都与容量相关，但在大多数情况下是这样的。当有足够的容量时，系统会正常工作。当容量不足时，系统会发生崩溃，出现很多不同表现。

在DW2.0环境中技术员需要注意几种容量及相关的指标，包括：

- 内存，对所有类型的处理，尤其是交互环境下的在线事务处理
- 队列长度和容量（队列长度在系统中通常是一个瓶颈值）
- 缓存容量和命中率
- 硬盘空间
- 近线空间
- 归档空间
- 归档处理
- 网络容量
- 等等

通过观察以上各种指标，技术员能够在许多问题发生前就先行处理它。

还有其他一些重要指标，包括整合区中休眠数据的增加、近线存储的增长、归档存储的增长、整个环境中数据访问概率的测量、网络瓶颈，等等。简而言之，技术员在任何地方能提前避免重要的短缺问题都会更好。

管理终端用户的关系和期望是DW2.0环境下的一项非常重要的管理工作，如果管理员忽视了这一点，管理就会存在很大风险。终端用户期望的管理方式包括：

- 设立服务台。
- 定期发布针对如何使用DW2.0的包含成功案例和帮助提示的实时简讯。
- 偶尔在内部开设一些讲述DW2.0环境各个方面的内容与使用的课程。
- 实行指导委员会，这样终端用户就可以决定优先权和进度，或至少给出一些意见。
- 让终端用户参与DW2.0环境完整的设计和开发周期。
- 实行一体的“展示和讲述”会议，并由此实行内部会议。
- 偶尔让外部专业人员参加短期研讨会，以补充DW2.0的经验和信息。

SLA（即服务标准协议）的建立，也是管理终端用户关系的重要部分，SLA是在DW2.0中日常的处理中测量的。SLA提供了一个可度量的开放的系统性能记录。建立SLA对终端用户和技术员都有帮助。通常，SLA同时解决了在线性能和可用性的问题。另外，分析环境中使用的SLA和事务环境中使用的SLA有很大的不同。

有时候在DW2.0中需要进行统计处理，此时技术员必须仔细监视统计处理对资源利用的影响。到一定程度时，需要建立单独的设备来研究统计分析。

23.7 DW2.0环境管理人员的管理

管理人员涉及所有的管理活动，其任务就是保证满足管理DW2.0环境的各个目标和目的。如下是其中一些重要的方面。

23.7.1 优化及优先冲突

当面临优化问题时，人们就都会出现在经理办公室中。几乎总是会出现这样的情况，即一些部门要对 DW2.0 进行修改和添加操作，与此同时另一个部门也要对其进行修改和添加操作。此时经理的工作就是解决（至少是改善）这些冲突。一些典型的考虑包括：

- 在 DW2.0 中添加哪些组件可以得到最大的财政回报。
- 在 DW2.0 中添加哪些组件最容易、最快。
- 在 DW2.0 中添加哪些组件可以在组织机构可接受的时间框架内完成。
- 在 DW2.0 中添加哪些组件可以得到最大的战略回报。

对于企业，当需要决定添加或修改的顺序时，管理人员必须仔细思考这些问题。此外，在管理 DW2.0 环境时还有其他要考虑的问题。

23.7.2 预算

预算是管理人员影响组织机构的主要方式。得到了资金的项目就可以继续进行，而没有得到资金的项目则无法继续进行。预算分为长期预算和短期预算。在 DW2.0 环境中，几乎所有的事情都是以迭代的方式完成的。这就意味着管理人员有机会做一些长期和短期的纠正，这也是预算过程中很正常的一部分。

23.7.3 进度表和里程碑的确定

里程碑和进度表的设置是管理人员工作中的一个重要部分。通常，管理人员并不创建最初的进度表和里程碑，而是让项目组提出进度表和里程碑。然后，管理人员批准这些可接受的进度表和里程碑。由于 DW2.0 的各方面几乎都是以迭代方式来构建的，管理人员也就有足够多的机会来影响整体的进度表。

23.7.4 资源分配

经理选择谁来领导项目是一门艺术。一种学派的观点是，当项目出现问题时，就投入更多资源。不幸的是，这会向组织机构传递一种错误的信息：一种能够得到更多资源的可靠方法就是让项目陷入麻烦中。还有一种方法是任何项目陷入麻烦时就解雇项目负责人。不幸的是，有很多合理的情况会使一个项目陷入麻烦。管理的艺术在于确定即将面临的情况，并做出合适的决定。换一种说法，就是管理人员要能够分辨出快速碾过减速带和掉下悬崖的区别。

23.7.5 管理咨询人员

由于缺少关于 DW2.0 中的开发技能，企业向外面的咨询人员寻求帮助是非常正常的。管理人员需要能够客观地挑选咨询公司，而不一定挑选那些首选的公司，原因是首选的公司可能没有任何经验。另外，管理人员需要警惕那些咨询公司，他们以能力为卖点，却为项目配备了一些新雇用的正在摸索经验的职员，这是以牺牲客户利益为代价的。有多种办法可确保咨询公司不向不知情的企业“出售货物”：

- 不要签署超过 12 个月的合同。假如这个咨询公司是值得雇用的，那么 12 个月后，如果工作圆满完成了就继续签署合同。相反，如果没有按照合同的规定很好地完

成工作，那么就再雇用新的咨询公司。

- 确保有切实可行的短期交付物。这是判断是否真正取得进展的一个好方法。
- 确保咨询公司具体说明都有谁参与项目，关键职位是关于设计和管理工作的。
- 安置两三名企业职员负责项目的关键职位，与顾问一起手把手工作。这样一旦出现问题，企业职员自己能够判断是否需要通知管理人员。
- 将各种关键的设计都写成文档，并保证任何时间这些文档对管理人员都是可用的。
- 检查咨询公司的各项资格证明。不要仅仅因为咨询公司是一家大型的知名公司，就轻易地认为它一定能建立 DW2.0 环境。
- 警惕咨询公司禁止外来专家偶尔对工作进行审查。一家有信心、有实力的咨询公司会很乐意让其他专家进行审查，尤其是设计、开发、实施出现问题的时候。
- 警惕与硬件/软件供应商绑定在一起的咨询公司。咨询公司提出的建议常常是一种可察觉的供应商的产品。
- 公开与其他企业共享管理经验。如果其他管理人员公开讨论他们的经验，那么你可以从中学到很多东西。
- 警惕展示一个别的公司的经理的供应商。大多数情况下，这些经历都有一些你并不知道的安排。在某些情况下，咨询公司的经理事实上就是供应商的雇员，或至少是供应商的代理。
- 警惕供应商事先安排好咨询公司以达到他们自己的目的。很多软件供应商与咨询公司秘密“勾结”，你获得的评估结果很可能不正确。
- 警惕那些声称做产品套件的市场评估的公共顾问。这些顾问与供应商经常做一些秘密的安排，目的就是诱使你购买他们的产品，而不是给你一些诚实的产品市场评估。
- 警惕那些声称做市场研究和和产品评估的公司。你应该清楚，很多调查公司会向供应商出售一些服务，这会对供应商的产品评估产生影响。如果市场评估公司声明了他们花在市场调查以及产品评估上的费用，那么供应商的评估就是有效的。但如果市场调查公司隐瞒了对供应商进行评估的费用，那么市场调查公司所做的各项建议及评定就一定是不可信的。

23.8 总结

总之，DW2.0 环境的管理工作体现在很多方面，包括：

- 数据模型
- ETL 环境
- 数据库
- 管家
- 技术及系统
- 网络管理
- 归档处理
- 近线存储
- 交互处理
- 元数据管理